PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-041299

(43)Date of publication of application : 08.02.2002

(51)Int.Cl. G06F 9/45
G06F 9/44
G06F 12/00

(21)Application number : 2001-129924

(22)Date of filing : 26.04.2001

(71)Applicant : MICROSOFT CORP

(72)Inventor : BURD GARY S
COOPER KENNETH B
GUTHRIE SCOTT D
EBBO DAVID S
ANDERS MARK T
PETERS TED A

(30)Priority

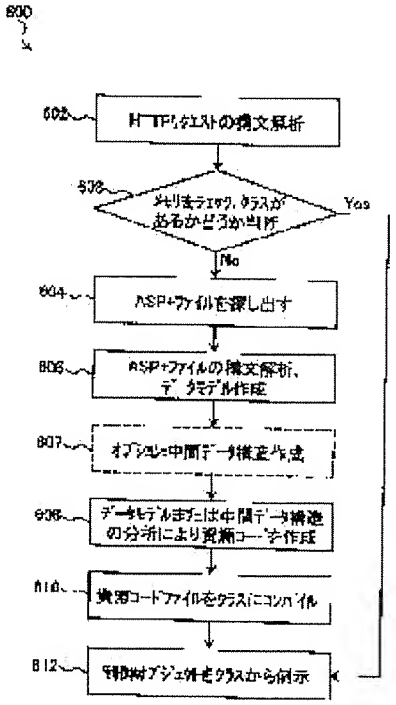
Priority number : 2000 573768 Priority date : 18.05.2000 Priority country : US

(54) GENERATION OF SERVER SIDE CODE FROM DYNAMIC WEB PAGE CONTENTS FILE

(57)Abstract:

PROBLEM TO BE SOLVED: To provide a development framework that can generate and process Web pages dynamically with developers minimum programming workload.

SOLUTION: A method generates an intermediate language or a resource code file from the server side resource or a dynamic Web page file, and the resource codes are compiled into an executable class that enables a quick generation of a Web page control object that performs the server side functions including rendering of client responses. The code generation scheme generates a control objects that is connected at a hierarchy in order to handle the event process and the



setting of attributes to a specific object.

LEGAL STATUS

[Date of request for examination]	11.12.2003
[Date of sending the examiner's decision of rejection]	
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]	
[Date of final disposal for application]	
[Patent number]	
[Date of registration]	
[Number of appeal against examiner's decision of rejection]	
[Date of requesting appeal against examiner's decision of rejection]	
[Date of extinction of right]	

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[Field of the Invention] Especially this invention relates to the server side code creation which creates the control object which generally processes the client side user interface element of a web page about a web server framework.
[0002]
[Description of the Prior Art] A typical web browser receives data from the web server which defines the appearance of a web page and basic actuation which are displayed on a client system. As a typical procedure, a user specifies the uniform resource (resource) locator ("URL") which is the global address of the resource on World Wide Web, and a desired website is accessed. An example of URL is "HYPERLINK "http://www.microsoft.com/ms.htm" http://www.microsoft.com/ms.htm." The 1st part of this example of URL shows the given protocol (for example, "http") used for a communication link. The 2nd part specifies the domain name (for example, "HYPERLINK "http://www.microsoft.com" www.microsoft.com") which shows the location of a resource. The 3rd part specifies the resource in a domain (for example, file called "ms.htm"). This is followed and they are a browser and HYPERLINK "http://www.microsoft.com" www.microsoft.com. The HTTP (hypertext transport protocol) request relevant to the example of URL for taking out the data relevant to the ms.htm file in a domain is generated. The web server which is acting as the host of the www.microsoft.com site returns reception, the demanded web page, or a resource to a client system by the HTTP response, and displays a HTTP request on a browser.
[0003] The "ms.htm" file of the above-mentioned example corresponds with the web page file containing the static HTML (HyperText Markup Language) code. HTML is a plane (plaintext) text authoring language used for the document (for example, web page) creation on World Wide Web. Since it is such, an HTML file is taken out from a web server by the client browser which changes the HTML code into a actual visual image or a voice component, and is displayed as a web page. In a client computer system, this process displays the content of a web page defined as the passed HTML file. A developer specifies the text and list which are displayed on a browser and by which formatting was carried out, form, a table, a hypertext link, an in-line image, voice, and background graphics using HTML. However, an HTML file is a static file which is not supporting dynamic generation of web page contents in essence. Web page contents are the HTML codes returned to a client for a display. Such dynamic processing is related with the server side application which generates the HTML code in advance of transmission as a result of a processing step rather than only transmits a predetermined code to a client browser.
[0004] In order to handle the dialogue between more complicated client-server, the server side application program has been developed so that the dialogue between more complicated client-server which gives dynamic contents, for example, the stock price which is changing, or traffic information may be handled. A server side application program processes a HTTP request, and generates the suitable HTML code for transmission to a client by the HTTP response to a client. For example, a server side

application program can process the enquiry string offered by the client side in a HTTP request, or the data from web based form, in order to generate dynamically the transmitting HTML code in the HTTP response to a client side. Intrinsically, server side application can generate the HTML mold file customized based on the information in the request from a client side. In this case, the static HTML file stored on the server does not exist, but an HTML file is dynamically created at the time of activation. The HTML code may be generated using the sequence of one or more text write-in processings to a memory device by which formatting was carried out as an example of a server side application program. Then, it is transmitted to a client system by the HTTP response, and the obtained text is displayed on a browser there.

[0005] Development of a server side application program is a complicated activity as which it is required it is not only well versed in the usual HTML coding used for a web page design, but that it is well versed in pro GURAMUBE six including one or more programming language (for example, C++, Perl, Visual Basic, or Jscript). However, although a web page architect is graphic designer or an editor in many cases and gives humane sensibility to a regrettable thing, it is inexperienced in programming in many cases. Therefore, it is required for a man with few programming experiences to offer the web page development framework simplified for creating the web page file which can develop the web page interface of server side application and each of its client. Therefore, to offer the development framework to which a developer can create and process a web page dynamically by the minimum programming is desired.

[0006]

[Problem(s) to be Solved by the Invention] One means which makes min the requirements for a program of dynamic web page generation is an active server page (ASP) framework offered by Microsoft Corp. By the ASP framework, a developer is Visual Basic or Jscript typically. The "ASP" web page file containing a code and the other HTML codes can be created. An ASP file contains the declaration or the tag and VB script, or J script code which performs various functions. It is easier to write in these declarations generally, rather than it writes in a actual programming code.

[0007] During processing, a HTTP request specifies an ASP file as a desired resource, and generates the HTML code after that using an ASP file as a result of the HTTP response to a client side. Furthermore, in order to reduce a given application programming effort, or the ASP file was developed beforehand, refer to a third party's client side library component (for example, client side "ACTIVEX" control) and a database, or other 3rd person applications for it.

[0008] The simplified ASP web page file must be changed into the script which may be interpreted with a script engine in the execution time. Typically, a script engine executes continuation or various declaration type commands [in / synchronous / an ASP file], in order to attain the result of a request. It is compiled as a file in which executive operation is possible, and, generally the script file performed with the script engine takes time amount as compared with the stored file. This is because a script engine not only performs a file, but must achieve an interpretation function.

[0009] Or one problem which is at the time of compiling a script file to the file in which executive operation is possible probably has the combination of various language, I hear that there is the possibility and it is in a script file. For example, a script file may contain other components written by the component written in HTML, and Visual Basic. Although various processings are used in order that a script engine may interpret these elements at the execution time, the compiler for translating a different language component into one language, i.e., one resource code file, does not exist. Furthermore, in the present server side application framework, programming required to manage dynamically client side user interface elements (for example, a text box, a list box, a carbon button, a hyperlink, an image, voice, etc.) within server side application needs a still advanced programming technique and considerable efforts. It becomes impossible for a script engine to be able to continue filling this demand continuously as these servers side process becomes more complicated.

[0010] This invention was made for these and other reasons.

[0011]

[Means for Solving the Problem] A resource code file is compiled by the class in which executive

operation is possible about the code generation approach for this invention to create intermediate language or a resource code file from a server side resource, and equipment. Quick generation of the web page control object which performs the server side function containing the rendering of a client response by the class in which executive operation is possible is attained. In an operation gestalt with this invention, a code generation scheme can create the control object connected within the hierarchy, in order to treat setting out of the attribute to event processing and a specific object. Furthermore, the code generation approach can also connect the object declared using the template again.

[0012] This invention relates to the creation approach of the class in server computer system memory more preferably. In order that a class may create the server side object which carries out the rendering of the web page contents dynamically, it is used according to a server computer system, and web page contents are sent to a client side computer system, and are displayed as a web page on a client computer system. In processing, a server computer system receives the request for a web page from a client computer system, and this request identifies a dynamic web page contents file. A server computer creates the data model for storing the element of a dynamic web page contents file, evaluates a data model, and generates the resource code file about a dynamic web page contents file based on assessment of a data model. If a resource code is created, a resource code file will be compiled and will create the compiled class in memory. Generally, a server computer system is enabled to return "refer to the class" to a server computer system, and to use a class for it, and it ends a process.

[0013] According to other desirable operation gestalten, an approach stores a class in the cache memory on a server computer system. If stored in cache memory, many server side page objects will be illustrated from one compiled class, and a resource from the first will not once be used again. It judges whether a server computer system has in memory the class by which the dynamic web page contents file was compiled whenever it received the request to a web page. It is created when the demanded class does not exist in memory. If there is a class, a server computer system will illustrate a server side processing object from the class, in order to generate web page contents dynamically. Next, the rendering of the web page contents is carried out, and they are sent to a client computer system.

[0014] According to still more nearly another operation gestalt of this invention, the approach step which evaluates a data model is accompanied by the repetitive traverse of a data model with two or more pass. A resource code is generated at the time of each pass, and it is written in a resource code file based on assessment of the data model at the time of the pass. A data model is built using the DS connected hierarchical.

[0015] According to a computer system, the operation gestalt of the computer program product by this invention can be read, and contains the computer program storage which codes the computer program which carries out executive operation of the computer process which creates the class compiled by memory on the server computer. The compiled class is used for illustrating a server side processing object in order to carry out the rendering of the response corresponding to the demanded web page which is displayed on a client computer system. Shape is taken by the subcarrier by the computer system and another operation gestalt of the computer program product by this invention includes the computer data signal which codes the computer program for creating the class compiled by the server.

[0016]

[Embodiment of the Invention] The gestalt of operation of this invention is related with the approach of generating the class compiled by the memory for the specific web page defined by the dynamic web page contents resource or the file. Creation of the compiled class is accompanied by creation of a resource code file from a web page file. Next, a class compiles a resource code file. When the compiled class exists in memory, it may be illustrated in order that a page object may carry out the rendering of the response returned to the client to display. Generally, a page object is accompanied by the server side control object for processing of the client side user interface element displayed on a web page, and generation. Furthermore, the hierarchy of a server side control object may be declared to the web page file which collaborates eventually and generates an authoring language code a result like HTML for the display of these objects of the web page on a client.

[0017] Drawing 1 shows the web server which generates dynamically the web page contents displayed

on the client in an operation gestalt with this invention. A client 100 carries out executive operation of the browser 102 which displays a web page 104 on the indicating equipment of a client 100. A client 100 includes the client computer system which has a display like a video monitor (not shown). The "INTERNET EXPLORER" browser currently sold by Microsoft Corp. is an example of the browser 102 in an operation gestalt with this invention. As an example of other browsers, it is "NETSCAPE NAVIGATOR". It reaches. Although there is "MOSAIC" etc., it does not restrict to this. The text box control 106 and two carbon button control 108 and 110 are included in the illustrated web page 104. A browser 102 receives the HTML code from a web server 116 by the HTTP response 112, and displays the web page described by the HTML code. Although HTML is explained with reference to a certain operation gestalt, especially, it is not restricted and it is thought that other authoring languages containing SGML (Standard Generalized Markup Language) and XML (eXtensible Markup Language) are within the limits of this invention.

[0018] The communication link with a client 100 and a web server 116 can be performed using the sequence of the HTTP request 114 and the HTTP response 112. Although HTTP is explained with reference to a certain operation gestalt, it is not restricted especially and it is thought that other transport protocols including S-HTTP are within the limits of this invention. In a web server 116, the HTTP pipeline module 118 analyzes reception and URL for the HTTP request 114, and calls the suitable handler 120 which processes a request. The web server 116 is equipped with two or more handlers 120 treating the resource of a different type in the operation gestalt with this invention.

[0019] For example, when URL specifies a static contents file 122 like an HTML file, a handler 120 accesses the static contents file 122, and sends the static contents file 122 to a client 100 by the HTTP response 112 through the HTTP pipeline 118. Or in an operation gestalt with this invention, when it specifies a dynamic contents resource or a file 124, a handler 120 accesses the dynamic contents file 124, processes the contents of the dynamic contents file 124, and generates the HTML code the result for web page 104. [as / whose URL is an ASP+ (Active Server Page+) page] Generally, a dynamic contents resource like a file 124 is the server side declaration data storage section which can use the authoring language which describes the web page which should be displayed on a client for generating dynamically. And the HTML code for web pages passes along the HTTP pipeline 118, and is sent to a client 100 by the HTTP response 112.

[0020] During this processing, the handler 120 was beforehand developed again, in order to simplify a development effort, or it can access the library of the 3rd person code. One of such the libraries is the server side class control library 126, and a handler 120 can illustrate from here the server side control object which generates HTML data as a result of processing a user interface element and displaying on a web page. in an operation gestalt with this invention, one or more server side control objects are visible on the web page described by the dynamic contents file 124 -- it hides-like and maps to one or more user interface elements.

[0021] A handler 120 accesses one or more non-user interface server components 130 which carry out executive operation on a web server 116 or another accessible web server again. A non-user interface server component 130 like stock price retrieval application or a database component is referred to in the dynamic contents file 124 processed by the handler 120, or is related with it. The non-user interface server component 130 can process the event started by the server side control object declared by the dynamic contents file 124. Consequently, the processing offered by the server side control object can simplify programming of the non-user interface server component 130 by encapsulating processing and generation of a web page of a user interface element, and, thereby, the developer of the non-user interface server component 130 can be concentrated on development of the function of the application proper instead of a user interface problem.

[0022] Drawing 2 shows the flow chart of processing of the client side user interface element using the server side control object in an operation gestalt with this invention, and generation processing. In processing 200, a client transmits a HTTP request to a server. A HTTP request contains URL which specifies resources, such as an ASP+ page. In processing 202, a server calls the suitable handler which receives a HTTP request and processes the specified resource. Reading appearance of the ASP+ page is

carried out in processing 203. Processing 204 generates a server side control object hierarchy based on the content of the specified dynamic contents file (for example, ASP+ page).
[0023] In processing 206, a control object hierarchy's server side control object performs postback event handling, postback data handling, status management, and one or more processings in data coupling. In processing 208, in order that each server side control object in a hierarchy may generate data like the HTML code for the display by the web page of a client side user interface element (or rendering), it is called. Although the vocabulary "a rendering" may mean the processing which displays graphics on a user interface, in this description, the vocabulary "a rendering" also means generation processing of the authoring language data which may be interpreted by client application like the browser for a display and a client side function. More detailed explanation of processing 206 and the rendering processing 208 is given in connection with drawing 6 . The call of the render method in each control object is performed using a tree traversal sequence. That is, the call of the render method of a page object becomes the repetitive traverse covering the suitable server side control object in a hierarchy.
[0024] Or actual creation of each server side control object may delay until a server side control object is accessed in processings 206 or 208 (handling of a postback input, loading of a condition, rendering of a control object to the HTML code, etc.). When a server side control object is not accessed for a given request, server processing is optimized by delaying creation of a control object and eliminating unnecessary control object creation processing.
[0025] In processing 210, the HTML code is transmitted to a client by the HTTP response. In processing 214, a client receives the HTML code relevant to the new web page which should be displayed. In processing 216, a client system displays the user interface element of a new page according to the HTML code received from the HTTP response. A server side control object hierarchy is ended in processing 212. The server side control object in a hierarchy answers the HTTP request which refers to the associated ASP+ page, is created, and after the rendering of authoring language data (for example, HTML data) finishes, it is destroyed. Or processing 212 may be performed before processing 210 after processing 208.
[0026] Drawing 3 shows an example of the module in the web server used in an operation gestalt with this invention. A web server 300 receives the HTTP request 302 to the HTTP pipeline 304. The HTTP pipeline 304 may also contain various modules, such as logging of web page statistics, user collating, a right of user access, and a module for the formation of an output cache of a web page. Eventually, each input HTTP request 302 received by the web server 300 is processed by the specific instance of an interface handler (Interface Handler) (it illustrates as a handler 306), for example, an IHTTP handler class. A handler 306 analyzes a URL request and calls a suitable handler factory (for example, page factory module 308).
[0027] In drawing 3 , the page factory 308 related with the ASP+ page 310 is called, and the instantiation and configuration of an object from the ASP+ page 310 are handled. Although the ASP+ page 310 may be recognized or referred to by URL of a meaning and other prefixes can be used for it by it, it is further discriminable with the prefix ".aspx", for example. If the request to specific ".aspx" resource is first received by the page factory module 308, the page factory module 308 will search a file system, and will obtain a suitable resource or a suitable file (for example, .aspx page 310). This file may also contain the text (for example, authoring language data) which may be behind interpreted or accessed by the server which processes a request, or the data (for example, cutting tool code data or coded data) in another format. When a physical file exists, the page factory module 308 reads a file to an aperture, and reads the file to memory. Or although the demanded file exists, when beforehand loaded to memory, a resource may not need to be loaded to memory necessarily so that it may state to a detail below. When the demanded aspx file is not found, the page factory module 308 returns the suitable error message "a file is not found" by transmitting for example, HTTP"404" message to a client.
[0028] After reading the ASP+ page 310 to memory, the page factory module 308 processes a file content, and builds the data models (for example, a list [of script blocks], directive, static text area, and hierarchy server side control object, a server side control property, etc.) of a page. A data model is used for generating the source code file of a new object class like the COM+ (Component Object Model+)

class to which the class of the page base which is the code which defines the structure, property, and function of a page object is made to extend. In an operation gestalt with this invention, a source list is dynamically compiled by intermediate language. It is behind compiled by the instructions (for example, X86, Alpha, etc.) of a proper on a plat form timely (Just-In-Time). Intermediate language is COM+ IL. A code and Java A cutting tool code and Modula 3 A code and SmallTalk A general purpose or custom-made orientation linguistic code of a code, the Visual Basic code, etc. may also be included. In another operation gestalt, intermediate-language processing is excluded and the instruction of a proper is directly generated from a source list or a source file (for example, ASP+ resource 310). The control class library 312 may be accessed with the page factory module 308, in order to obtain the server side control class which is used for a control object hierarchy's generation and which was defined beforehand.

[0029] The page factory module 308 creates a page object from the compiled class. The page object 314 is a server side control object equivalent to the web page 104 of drawing 1 . The page object 314 and its child object (for example, the text box object 318, the carbon button object 320, and another carbon button object 322) are the control object hierarchy's 316 examples. The example of other control objects can be considered according to this invention, like the custom-made control object, is not restricted especially and also contains the object (after-mentioned) corresponding to the HTML control shown in a table 1. The page object 314 corresponds to the web page 104 of drawing 1 . The text box object 318 corresponds to the text box 106 of drawing 1 . Similarly, the carbon button object 320 corresponds to the additional carbon button 108 of drawing 1 , and the carbon button object 322 corresponds to the deletion carbon button 110 of drawing 1 . The page object 314 relates to other control objects and hierarchy targets on a server. In a certain operation gestalt, a page object is a container object which contains the child object hierarchical. The hierarchical relationship of other gestalten, such as a dependency, can be used with another operation gestalt. In the more complicated control object hierarchy who has the child object of a large number level, one child object may be a container object of other child objects.

[0030] It sets in the above-mentioned operation gestalt, the control object hierarchy's 316 control object is created and performed on a server 300, and each server side control object corresponds to the corresponding user interface element and corresponding logic target on a client. A server side control object collaborates, handles the postback input from the HTTP request 302 again, manages the condition of a server side control object, performs data coupling with a server side database, and generates the authoring language data (for example, the HTML code) used for the display of a web page as a result of a client. Result authoring language data are generated from the server side control object hierarchy 316 (namely, rendering), and are transmitted to a client by the HTTP response 324. For example, in addition to this, control of an ACTIVEX type or when it is processed by the browser, refer to the HTML configuration which produces client side user interface elements (for example, a control carbon button, a text box, etc.) for the result HTML (or other authoring languages) code.

[0031] By declaration made with the ASP+ resource 310, a server side control object can access one or more non-user interface server components 330 for a dialogue with the non-user interface server component 330 and a client side user interface element. For example, a postback input can be answered and a server side control object can start a server side event to the non-user interface server component registered into those events. Thus, the non-user interface server component 330 minds a user interface element for a dialogue with a user, and it can perform it, without programming a code required displaying and processing these elements.

[0032] Drawing 4 shows the content of an example of the dynamic contents resource in an operation gestalt with this invention. In the illustrated example, the file 400 includes plain text declaration in a certain dynamic contents file format (for example, ASP+). Each declaration gives an instruction to the page factory module 308 which reads a file 400, creates a class, and calls the suitable server side control object which, in addition to this, carries out the rendering of the HTML code or the authoring language which transmits to a client by the HTTP response eventually.

[0033] the 1st line of a file 400 -- the following formats -- setting -- a delimiter -- :<%@ directive which contains a directive between "<%@" and " %>" [attribute=value] %> -- it is here, and especially directive is not restricted and may also contain "page", "cache", or "import". In order to determine

buffering semantics, the requirements for a session condition, an error handling scheme, scree PUTINGU language, transaction semantics, and a property like an import directive, a directive is used with the page factory module 308, when processing a dynamic contents file. A directive may exist anywhere in a page file.
[0034] <html> of the 2nd line is a standard HTML initiation tag written in a resource code file as a literal so that additional processing may not be performed in the information for carrying out the rendering of the result HTML code except a direct "write-in" instruction. In HTML functor, <html> shows the beginning of an HTML file and has become the termination tag </html> of the line 21 this [whose] is also a literal, and a pair.
[0035] A code declaration block exists in the lines 3-10 of a file 400. Generally, a code declaration block defines the method by which executive operation is carried out on a page object, a control object member variable, and a server. In the following formats : <script runat = "server" [language = "language"] and [src = "externalfile"] > ..... </script> Here, language and a src parameter are arbitrary. In an operation gestalt with this invention, a code declaration block is defined using the <script> tag including the "runat" attribute which has the value set as the "server." Since the syntax of an inner code is specified, a "language" attribute may be used for arbitration. Although default language can express the language configuration of a whole page, a developer is Jscript by the "language" attribute of a code declaration block. And different language within the same (Practical Extraction and Report Language) web page activation of PERL etc. can be used. The <script> tag can specify a "src" file as arbitration again, and a "src" file is a file of the exterior where a code is inserted in the dynamic contents resource for processing by the page compiler from there. Although the syntax currently indicated is used in this operation gestalt, with another operation gestalt, he should understand that different syntax within the limits of this invention can be used.
[0036] In drawing 4 , two subroutines, AddButton#Click, and DeleteButton#Click are declared in the Visual Basic format into the code declaration block. It will be called, if any subroutine takes two input parameters, "Source", and "E" and a client side click event is detected by the carbon button of a response. In an AddButton#Click subroutine, the text in a user name text box is connected with word "Add", and is loaded to the text data member of a message. In a DeleteButton#Click subroutine, the text in a user name text box is connected with word "Delete", and is loaded to the text data member of a message. Although not shown in drawing 4 , the member variable of a server side control object may be declared with the code declaration block of a file 400. For example, if Visual Basic syntax is used, keyword"DIM" will declare the data variable of a server side control object.
[0037] "A code rendering block" (not shown) may be included in a dynamic contents resource. A code rendering block can contain the code of the amount of the arbitration by which executive operation is carried out by page rendering time amount. In an operation gestalt with this invention, executive operation of the code rendering block is carried out by one "rendering" method by which executive operation is carried out at the time of a page rendering. Executive operation of other parts of a code may be carried out by the rendering method. A code rendering block fills the following formats (other formats are considered in another operation gestalt).
[0038] <% InlineCode %> "InlineCode" includes the independent language mold code block or flows-of-control block which carries out executive operation on a server at the time of a page rendering here.
[0039] in-line one -- an expression -- again -- being as follows -- instantiation ---like -- syntax -- using -- an expression -- a rendering -- a block -- a delimiter -- " -- < -- % -- @ -- " -- " -- % -- > -- " -- between -- it can use .
[0040] <%= InlineExpression %> Here, the expression included in the "InlineExpression" block is eventually included by the call to "Response.Write (InlineExpression)" of the page object which writes the value from "InlineExpression" in the holder of the suitable location in declaration. For example, "InlineExpression" may be contained in a file 400 as follows.
[0041] <font size = "<%=x%>" > Hi <%=Name%>, you are <%=Age%>! This outputs a greeting and the description about a certain man's age with the font stored in value"x." The man's identifier and age are defined as a string in a code declaration block (not shown). The rendering of the

result HTML code is carried out by the server, and it is transmitted to a client by the HTTP response so that the value of "InlineExpression" may be included in a suitable location. That is, it is as follows.
[0042] <font size ="12"> Hi Bob and you are 35! In the line 11 of a file 400, <body> is a standard HTML tag for specifying the beginning of the text of a HTML document. The termination tag </body> is shown in the line 20 of FIIRU 400. In an operation gestalt with this invention, both <body> and </body> are literals.
[0043] The initiation tag <form> of a HTML foam block is seen on a line 12 in this section of the file 400 of drawing 4 . The termination tag </form> of a foam block is seen on the line 19 of HTML file 400. Since a given identifier is related with a foam block, parameter "id" of arbitration can also be included in a HTML control tag, and it enables this to contain many foam blocks in one HTML file.
[0044] The server side label identified by the "message" is declared in the line 18 of a file 400. A "message" label is used in code declared with the lines 5 and 8 of a file 400, in order to display a label on a web page.
[0045] In the foam block, three examples of a HTML control tag corresponding to the user interface elements 106, 108, and 110 of drawing 1 are shown. The 1st user interface element is declared with the line 13 of the file 400 equivalent to a text box. Text literal "User Name" declares the label located in the left-hand side of a text box. The input tag which has type="Text" declares the text box server side control object which considers as the server side control object which carries out the rendering of the text box client side user interface element, and has an identifier called "UserName". The lines 15 and 16 of a file 400 declare the client side user interface element shown as carbon buttons 108 and 110 of drawing 1 , respectively. A "OnServerClick" parameter specifies the suitable subroutine declared with the code declaration block of a file 400. And the server side carbon button control object generated by the response to declaration by the file 400 carries out the rendering of the server side code of the relation which performs the HTML code and carbon button click event of a client side carbon button.
[0046] The text box and carbon button which were declared by the file 400 are the example of HTML server control declaration. In an initial state, the HTML tags in an ASP+ resource are dealt with [no] as literal text contents, and can be accessed in a page developer's programming. However, by specifying using the "runat" attribute which has the value set as "server", the syntax of a HTML tag is analyzed and a page developer can show that it should be treated as accessible server control declaration. Each server side control object can be related with the "id" attribute of the meaning which enables program reference of a corresponding control object at arbitration. The property argument on a server side control object and event association can also be specified using the declaration name / value attribute pair on a tag element (for example, OnServerClick is equal to a "MyButton#Click" pair).
[0047] In an operation gestalt with this invention, the general syntax which declares a HTML control object is as follows.
[0048]
<HTMLTag id = "Optional Name" runat = server> ..... </HTMLTag> Here, "Optional Name" is the identifier of the meaning of a server side control object. Although the list, the related syntax, and COM+ class of a HTML tag which may be supported are shown in a table 1, other HTML tags can be considered within the limits of this invention.
[0049]
[A table 1]

HTML タグ名	例	COM+ クラス
<a>	 My Link 	AnchorButton
<img>	<img id = "MyImage" runat = server>	Image
		Label
	Some contents	Panel
<form>	<form id = "MyForm" runat = server> </form>	FormControl
<select>	<select id = "MyList" runat = server> <option>One</option> <option>Two</option> <option>Three</option> </select>	DropDownList
<input type = file>	<input id = "MyFile" type = file runat = server>	FileInput
<input type = text>	<input id = "MyTextBox" type = text>	TextBox
<input type = password>	<input id = "MyPassword" type = password>	TextBox
<input type = reset>	<input id = "MyReset" type = reset>	Button
<input type = radio>	<input id = "MyRadioButton" type = radio runat = server>	RadioButton
<input type = checkbox>	<input id = "MyCheck" type = checkbox runat = server>	CheckBox
<input type = hidden>	<input id = "MyHidden" type = hidden runat = server>	HiddenField
<input type = image>	<input type = image src = "foo.jpg" runat = server>	ImageButton
<input type = submit>	<input type = submit runat = server>	Button
<input type = button>	<input type = button runat = server>	Button
<button>	<button id = MyButton runat = server>	Button
<textarea>	<textarea id = "MyText" runat = server> This is some sample text </textarea>	TextArea

[0050] In addition to a standard HTML control tag, an operation gestalt with this invention enables a developer to create the reusable component encapsulated about a program function common to the outside of a HTML tag set. These custom-made server side control objects are specified using the declaration tag in a page file. Custom-made server side control object declaration includes the "runat" attribute which has the value set as "server". In order to carry out possible [of the program reference of a custom-made control object], the "id" attribute of a meaning is specified as arbitration. Furthermore, the declaration name / value which is an attribute pair in a tag element specify the property argument of a server side control object, and event association. An in-line template parameter may also be combined with a server side control object by giving the element of the child who is the prefix-letter train of a suitable "template" to a parent server control object. A format of custom-made server side control object declaration is as follows.

[0051] <servercntrlclassname id="OptionalName" [propertyname="propval"] runat=server/> Here, "servercntrlclassname" is an accessible control class, "OptionalName" is the identifier of the meaning of a server side control object, and "propval" expresses the property value of the arbitration of a control object.

[0052] An XML tag prefix can be used for offering the briefer notation for specifying a server side control object in a page by using the following formats using another specification statement method.

[0053] <tagprefix:classname id = "OptionalName" runat = server/> Here, in "tagprefix", in relation to a given control name tooth-space library, "classname" expresses the identifier of control in a relation name tooth-space library. "propertyvalue" of arbitration is supported.
[0054] If drawing 5 is referred to, an example of the computer system of the operation gestalt of this invention contains the general purpose computer equipment of the gestalt of the conventional computer system 500 containing the system bus 506 which connects the various system components containing the processor unit 502, a system memory 504, and a system memory 504 to the processor unit 500. System buses 506 may be any of the bus structure of some types containing the peripheral bus and local bus which use a memory bus or a memory controller, and various bus architectures. The system memory contains the memory (ROM) 508 only for playbacks, and random access memory (RAM) 510. The unformatted input / output system 512 (BIOS) containing the basic routine which helps a transfer of the information between the elements within a computer system 500 are stored in ROM508.
[0055] The computer system 500 contains the optical disk drive 518 which performs read-out and the writing of a removable optical disk 519 like the magnetic disk drive 514 and CD ROM which perform further read-out and the writing of a hard disk drive 512 and the removable magnetic disk 516 which perform read-out and the writing of a hard disk, DVD, or other optical media. The hard disk drive 512, the magnetic disk drive 514, and the optical disk drive 518 are connected by the hard disk drive interface 520, the magnetic disk drive interface 522, and the optical disk drive interface 524 system bus 506, respectively. A drive and its medium which can be related computer read offer the instruction of a computer system 500 which can be computer read, DS, a program, and the non-volatile storage section of other data.
[0056] Although the hard disk, the removable magnetic disk 516, and the removable optical disk 519 are used for this description in the above-mentioned environmental example of a publication, the medium type [other] in which data storage is possible and which can be computer read can be used for the above-mentioned example of a system. The medium of other types of these which can be used for the above-mentioned example of operating environment which can be computer read has for example, a magnetic cassette, flash memory card, a digital videodisc, a BERUNUI (Bernoulli) cartridge, random access memory (RAM), the memory (ROM) only for playbacks, etc.
[0057] Many program modules are stored in a hard disk, a magnetic disk 516, an optical disk 519, and ROM508 or RAM510, and these contain the application program 528, other program modules 530, and the program data 532 of 526 or 1 or more operating systems. A user can input a command and information into a computer system 500 with input units, such as a keyboard 534 and a mouse 536, or other pointing equipments. As other input devices, there are a microphone, a joystick, a gamepad, a satellite dish, a scanner, etc., for example. These and other input devices are connected to the processor 502 through the serial port interface 540 connected to the system bus 506 in many cases. However, these input devices may be connected by other interfaces, such as a parallel port, a game port, or Universal Serial Bus (USB), again. The monitor 542 or the indicating equipment of other types is also connected with the system bus 506 through the interface of a video adapter 544 etc. In addition to a monitor 542, typically, a computer system contains other circumference output units (not shown), such as a loudspeaker and a printer.
[0058] A computer system 500 can operate in the environment using the logical connection to one or more remote computers like a remote computer 546 connected by network. a remote computer 546 -- a computer system, a server, a router, Network PC, and a pier (peer) -- it may be equipment or other common network nodes, and there are many elements typically mentioned above in connection with a computer system 500, or all are included. Network connection contains Local Area Network (LAN) 548 and Wide Area Network (WAN) 550. Such a network environment is not new in office, an enterprise magnitude computer network, intranet, and the Internet.
[0059] When using by the LAN network environment, a computer system 500 is connected to a local network 548 through a network interface or an adapter 552. When using by the WAN network environment, a computer system 500 includes the modem 554 or other means for establishing the communication link by Wide Area Network 550 like the Internet typically. Built-in or external any is

sufficient as a modem 554, and it is connected with the system bus 506 through the serial port interface 540. In the environment connected by network, the program module described in relation to the computer system 500 or its part may be memorized by remote memory storage. The illustrated network connection is an example and can use other means for the communication link establishment between computers.

[0060] In the operation gestalt of this invention, a computer 500 expresses a web server and CPU502 carries out executive operation of the page factory module on the ASP+ file memorized by at least one of storages 516, 512, 514, 518, and 519 or memory 504. A HTTP response and a request communicate by LAN548 connected to the client computer 546.

[0061] The processing 600 performed with the page factory module 308 is shown in the flow chart of drawing 6 . In drawing 6 , processing 600 is mostly equivalent to generation processing (drawing 2) of the control object hierarchy 204. Processing 600 is changed into the object code class which had the ASP+ page (it is also called an ASP+ file) compiled, and subsequently, it is loaded to memory, in order to illustrate the control objects 314, 318, 320, and 322 (drawing 3).

[0062] If a server receives the request to URL in processing 202 (drawing 2), the page creation processing 600 will start. The syntax-analysis processing 602 analyzes syntax of demanded URL after reception of a request, and it judges which resource is demanded. Including files (.htm, .gif, .jpg, etc.) with a static resource, directory browsing starting, a DAV (digital audio/video) file, and dynamic contents requests (.aspx, .soap, etc.), this invention is constituted so that these may be handled. The HTTP request to which close [which was received by the server side] comes is eventually processed by the specific instance of a handler class, for example, an IHTTPHandler class. The architecture which interprets the URL request to a handler instance, for example, an IHTTPHandler instance, actually by the handler factory, for example, the activity of IHTTPHandler "a factory", and in which a spigot is possible is offered. Close can perform this interpretation easily using application configuration setting out which can map the file extension and the HTTP command of URL by which it comes to a factory class like the IHTTPHandlerFactory class which is to create eventually a suitable handler instance, for example, an IHTTPHandler instance. In syntax-analysis processing, although various resources are discriminable with spigot possible architecture, the following publications focused on the HTTP request which identifies a dynamic web page contents resource, and have focused on the .aspx or ASP+ page or a dynamic web page contents file like a file especially.

[0063] If a actual resource is identified by the syntax-analysis processing 602, in the check processing 603, memory will be checked and it will judge whether a class exists in memory. In order to judge whether the class is compiled or not, in the check processing 603, the class in memory is searched by searching the flag which will be set if a class is compiled, or the identifier. Whichever it makes it, in the check processing 603, the index which shows that a compile class is already compiled and is stored is searched. When a class is in memory, a flow branches on a YES branch and progresses to the instantiation processing 612. Instantiation processing illustrates a control object from the compiled class. In the check processing 603, if it is judged that the class is not compiled, a flow will branch on NO branch and will progress to the fixing processing 604.

[0064] If the check processing 603 judges that a class does not exist in memory, in the fixing processing 604, it will search and discover a specific resource and will read a file to memory. A base class like a "System.ASP WebForms.PageFactory" class offers the handler factory implementation which handles instantiation and configuration of ASP and an ASPX page. If a resource and a dynamic web page contents file physical in this case are not found, a suitable error message is returned.

[0065] Next it is the fixing processing 604, in processing 600, syntax-analysis creation processing 606 is performed and syntax of a resource like an ASP+ file is analyzed here. Syntax-analysis/creation processing 606 creates a data model from the information collected while reading and analyzing the syntax of an ASP+ file for every declaration. A data model is the DS containing the element relevant to the element of the active content file which can pull out a resource code. Including the structural element with which the data model was referred to by the active content file, these elements are connected so that the structure of a control object may be expressed as a result of an active server page. In an

operation gestalt with this invention, a data model is the combination of the object associated by the hierarchical tree structure.
[0066] Each declaration of a web contents file has the predetermined element in which the information stored in the creation approach of a data model and a data model is shown. For example, when declaration is literal text declaration, a data model should just include information by which a text is inserted in a suitable location. However, when declaration shows that a nested control object exists, the data model according to a nest which was declared to the ASP+ file must be created. Intrinsically, as for a data model, as for each object, a child object includes the information about whether it exists or not in each object including one object for every declaration.
[0067] If the syntax of an ASP+ file is analyzed and a data model is created, arbitrary creation processings 607 may be performed. The creation processing 607 creates the medium DS which abstracts a resource code like the resource code file which should analyze a data model, for example, should be created so that it may indicate below. Intrinsically, medium DS is comprehensive DS which describes a code. Medium DS has the DS of the upper level which describes a class. You may be the DS of another level which has the list or array of a class name and a method for every class. Furthermore, it is the DS which has each method itself, a statement, and various elements like declaration. A statement may include items, such as an expression and a data call.
[0068] If the syntax of an ASP+ file is analyzed and a data model (and arbitration medium DS) is created, typically, the creation resource code module 608 will create a required resource code through the analysis of each data model by writing in the various lines of the code of each declaration which exists in an ASP+ file. Generally, this step is accompanied by the well-known declaration in a data model, and direct translation in the code of other information. The hard coding of such a translation is carried out to the application which translates, or each translation may be stored in a retrieval table. Although this kind of retrieval table may be created so that the suitable translation for almost all language may be offered, the compile to COM or the COM+ class for using existing COM or an existing COM+ library becomes easy by the activity of object-oriented resource code language. (For example, there is C++ or Vbasic as an example of resource code language.) Further, when medium DS is created by processing 607, a translation becomes still more direct. In such a case, DS may become very close to a comprehensive resource code language file, and a translation may become the addition of an only suitable lexicon to create the resource code file of specific resource code language, and syntax.
[0069] An ASP+ file declares specific resource code language to a result resource code file. When language is not declared to an ASP+ file, default language, such as Visual Basic, can be used. As a result of being generated from a data model or medium DS, a resource code file is a resource code file with the advanced type which will be intrinsically required to write in when not receiving the benefit of this code generation tool by the case where a programmer uses a server side control object hierarchy.
[0070] Completion of a resource code file creates the class which compiled the resource code file and was compiled in object code or other executive operation possible forms in the compile processing 614. Or a class can be compiled in the language of the made others started on a cutting tool code or a virtual unit. Compile of a resource code file is accompanied by the activity of the compiler constituted so that the resource code language which exists in a resource code file might be compiled. And a class is called by the instantiation processing 616 and creates a control object as a result of the page object for a web page. Although not clearly equipped in a URL request, instantiation processing is un-explicit processing which is a part of URL request of a specific resource.
[0071] A page class's compile of an ASP+ file maintains a class in the usable condition by the future request. therefore, generation processing of the resource code of an ASP+ file is performed once -- sufficient -- a next request can use the compiled class and illustrates a required object if needed. As for the compiled class, the between cache of the short period of time may be carried out, and/or the compiled class can be stored in a disk. Once an ASP+ file is compiled, it is not actually necessary to touch an ASP+ file from the first again. Of course, when an ASP+ file is corrected, a page class is updated by repeating compile processing. It can judge whether the ASP+ file was updated using a suitable flag or other index mechanisms, therefore renewal of an automatic class is possible. Or after

updating ASP+ to a web page file, the charge of removing the class by which the cache was carried out may be left behind to a developer.

[0072] The detail relevant to creation of a data model is shown in drawing 7 . The data model creation processing 700 begins from judging whether in an ASP+ file, there is any declaration by the test processing 702. When there is no declaration like a pure HTML file which should be compiled, a flow branches on NO branch and progresses to termination of processing 714.

[0073] If it is judged that an ASP+ file has at least one declaration by the test processing 702, a flow will branch on a YES branch and will progress to the acquisition processing 704 which obtains the 1st declaration by the ASP+ file. The information from the 1st declaration is stored in DS by the storing processing 706. Not only the actual declaration stored in DS but the information relevant to the 1st declaration of whether whether control of the container type with which declaration has a child object being declared, and declaration declare a directive etc. may be stored in DS. Since a web page is an ASP+ file, it has an directive tag as the 1st declaration typically. a directive -- a tag -- versatility -- a directive -- for example, -- drawing 4 -- a line -- one -- being shown -- as -- a delimiter -- " -- < -- % -- @ -- " -- " -- % -- > -- " -- between -- information -- containing . The delimiter shown in drawing 4 is instantiation-like only, and should understand that other selections are possible. A directive supplies information to the page factory module used for resource code file creation. For example, the directive shown in the line 1 of drawing 4 shows that the default language which should be used is VB (namely, Visual Basic) in this example.

[0074] If the 1st declaration is taken out and it is stored in DS, processing 700 will progress to the test processing 708 which judges whether a file has another declaration. This processing is the same as the above-mentioned processing 702. If there is already no declaration, a flow will branch on NO branch and will progress to the termination step 714. If there is another declaration, a flow will branch on a YES branch, will progress to the acquisition processing 710, and will gain the next declaration.

[0075] It is the ejection of the next declaration, next the storing processing 708 stores the information relevant to the next declaration in different DS related with other DS of a data model. As the 1st declaration was described in the top, the information stored in DS may include not only the literal text of the next declaration but the information derived from the literal text. Therefore, DS may be related with the 1st DS hierarchical if needed.

[0076] Next it is storing of the information on the following DS, a flow branches to the test processing 708 and it judges whether the declaration which should be carried out reading appearance to a resource file, for example, an ASP+ file, at a data model still exists. When it does not exist, a flow branches on NO branch and progresses to termination of processing. however . If the test processing 708 judges that declaration still exists, a flow will branch on a YES branch and will progress to the acquisition processing 710 which gains the next declaration. Next it is the acquisition processing 710, in the storing processing 708, the information relevant to the next declaration is stored as mentioned above. These steps 708, 710, and 712 continue until all declarations are taken out one by one and stored in DS. DS may be associated as a hierarchy connected so that they can identify the sub element of a nested object, a child object, a node, or a parent object as a sub element.

[0077] In creation of the resource code module 608 (drawing 6), the resource code file of specific resource code language is created using this data model. It is important for creation processing of the resource code file from a data model to note being dependent on resource code language. That is, the resource code file written in in specific language has typically the specific requirements for a format, for example, the requirements that all adjustable declarations precede the activity of these variables. Therefore, if the compiler of the language is called, suitable program instruction must be arranged in the suitable location in a resource code file. In order to make and accomplish this activity, an ASP+ file must be evaluated, and it must judge of what kind of meaning an element or the description is in a file, and a file must be processed, and a resource code file must be generated. Since, as for a resource code file, the thing in beginning of a file typically like adjustable declaration of the element of the 1st lot or declaration is required, the whole data model should be analyzed for this information. Similarly, since there should be control object information in the center of a file, the whole data model should be

analyzed for this information at the time of the 2nd next phase of the 1st phase of processing.
[0078] These phases are functional processings which judge a part with the separate code which should be written in a resource code file (or medium DS in processing 607), and a separate part is together put logically here based on a location the result in a resource code file. Therefore, these phases are considered to be "pass" or traverse of one data model, therefore are performed continuously. however -- or these phases may be performed as separate concurrent processing, the result of separate processing is summarized in one resource code, and processing evaluates either of the separate copies of a single data model or a data model. Or the functional processing or the functional phase which judges and writes in a part with a separate code may be performed at the time of a single traverse, and a resource code is selectively written in a part with a separate resource code file, or is written in the separate file which was summarized so that the part might offer the suitable connection between the logical parts of a resource code, or was associated there. Therefore, in this description, although it has indicated substantially that it is carried out in succession by separate analysis one after another, the split of the analysis may be carried out simultaneously substantially, or it may be performed.
[0079] The processing or the flow shown in drawing 8 is related with the operation gestalt 607 which generates a direct resource code file comparatively from a data model, i.e., the abbreviation processing shown in drawing 6 . A variable is declared first and this operation gestalt generates a resource code file like the VB code file to which an object and a method follow variable information. The phase when the resource code creation processing 800 shown in drawing 8 is separate as "pass" of a data model is shown, and three times pass the DS of a data model. A variable and declaration information are looked for in processing of the 1st pass. In processing of the 2nd pass, object creation information is looked for and code rendering information is looked for in the 3rd pass processing 800.
[0080] The 1st pass starts in the processing 802 which traverses a data model about a variable and other declaration information. This step begins from analyzing the 1st DS of the data model created by processing 700. In analysis processing of the 1st DS, it judges whether there is any specific line of the resource code which should be written in the resource code file about declaration.
[0081] Termination of the analysis of the 1st DS generates and writes the resource code about a variable and declaration information in a resource code file in processing 804. A writer object is called and the text given to it is only written in a resource code file. Therefore, the call to a writer object contains both text strings of the syntax of the specific programming language about the parameter, the variable, or declaration information on a resource code file name. Or a text is copied or is written in a resource code file using other well-known means.
[0082] Moreover, the suitable syntax which should be passed to a writer object must be determined in processing 804. Intrinsically, when current DS expresses server side control, it generates the resource code to it in processing 804. Otherwise, in processing 804, it is copied to a resource code file. Intrinsically, processing 804 can perform one of three. In the generation processing 804, you may judge whether it follows [whether the information in an ASP+ file is a literal text, and] first, and should be directly written in a resource code file. This is the case where a HTML tag is inserted in an ASP+ file, and resource code declaration is not generated in this case. Instead, information is only copied to a resource file. In processing 804, you may judge whether 2nd the response of 1 to 1 is required in the suitable syntax form of the translation with a simple text, and specific resource code language. In such processing 804, when a hard code is carried out to a page factory module in search of 1 to 1 response, one-for-one translation is performed, and result information is supplied to a writer object. When one-for-one translation is not made [3rd], in processing 804, a more complicated translation is required, therefore a module or the next retrieval table processing is called, and creation processing of the suitable resource code syntax which should be passed to a writer object is performed. Typically, the translation of declaration information is direct.
[0083] In processing 804, if the 1st the generation and writing of a resource code of DS are performed, in the test processing 806, it will detect whether the DS which should be analyzed to a data model still exists. As for a flow, in a certain case, the DS which should be analyzed at the time of this 1st pass of a data model still branches on a YES branch, and it traverses to the following DS at the beginning of

processing 800 in return and 802. In such a case, the following DS is analyzed as mentioned above, generates resource code syntax, and calls the writer object which writes the syntax code in a resource file.

[0084] In the test processing 806, when it is judged that the DS which should be analyzed to a data model at the time of this 1st pass does not exist any longer, a flow branches on NO branch, progresses to the traverse processing 808, and starts the 2nd pass which passes DS. The traverse processing 808 starts the next phase of writing in the code which writes in an object fabrication and method creation typically. The traverse processing 808 begins from on a data model, and returns to the DS which a data model begins.

[0085] Next, the 1st DS is analyzed in generation and the write-in resource code processing 810, and it judges whether the information about object creation is in the 1st DS. When such object creation information exists in the 1st DS, it is transmitted to the writer object which the syntax of specific programming language is created and generated, and writes in a resource code file. Intrinsically, processing 810 is the same as the above-mentioned processing 804, if DS is analyzed about the information on a particular type and the information is collected from DS, will perform a certain translation and will create the resource code language about the information here. When information is a literal, in processing 810, the information can be directly sent to a resource code file. Similarly, when the information needs an easy translation, the easy translation may be a part of whether a hard code is carried out to a program, and retrieval table type processing. Furthermore, when complicated structure is required, a retrieval table or other modules are called, information is processed, and the suitable syntax resource code for a resource code file is created.

[0086] It is processing 810, next the enquiry step 812 tests whether the DS which should be analyzed to a data model exists. When the DS which should be analyzed for object creation exists in a data model, a flow progresses to the traverse processing 808 which branches on a YES branch and traverses the following DS. Therefore, in processing 808, the DS of the information about object creation is analyzed in return and processing 810 to the following DS. As mentioned above, it is transmitted to a writer object and this object creation information is translated into the resource code line which can be added to a resource code file. Steps 812, 808, and 810 are repeated until all resource codes are written in about object creation.

[0087] If the decision step 812 judges that the DS which should be analyzed for object creation does not exist in a data model any longer, a flow will branch on NO branch, and will progress to processing 814, and it will end the 2nd pass. This 3rd pass starts in the processing 814 of the top of a data model. As processings 808 and 802 were described in the top, this traverse step takes out the 1st DS in a data model first. It is the ejection of the 1st DS in the data model in processing 814, next the DS of code rendering information is analyzed by processing 816. It is transmitted to a writer or code rendering information is translated into the syntax [****] in the programming language made to add to a resource code file. A code rendering method is near the resource code end of file, therefore the 3rd pass is performed to the degree of object creation pass. Or a code rendering method is generated in between by simultaneous thread processing, and is added to a resource end of file.

[0088] Processings 816 may be collected from the above-mentioned processings 810 and 804 or the code rendering information which becomes and is alike and this resource code information has in DS. Furthermore, you may require as mentioned above that information should be inserted directly, should be translated at a rate of 1 to 1, and should generate the syntax of a resource code file using a more complicated module. Next it is processing 816, in the detection processing 818, it judges whether DS remains in the data model.

[0089] In processing 818, if it is detected as the DS which should be analyzed between this 3rd pass still existing, a flow will branch on a YES branch and will return to 814 for the traverse of the following DS. The following DS is called and the generation and the writing of a resource code about the code rendering information on the following DS are performed in processing 818.

[0090] In ** Li 818, if it judges that the DS which should be analyzed does not exist any longer, a flow will branch on NO branch and will end processing by the processing connection 820.

[0091] Several pass is performed by the above-mentioned explanation to a data model so that clearly. Each declaration of a data model depends this on requiring the specific line of the code put on the various locations of a resource code file. However, in case a resource code file is written in, a writer object can only perform adding to a file continuously and writing in it one by one, and cannot insert a resource code line between the code lines written in beforehand. Therefore, a data model must be traversed once or more, in order to collect the suitable information belonging to the part of the 3rd or the last of the 2nd or the pars intermedia of the 1st or the upper part of a resource code, and a resource code, and a resource code file. It is expected that the number of pass changes with number of the parts of the code which exists to various computer languages.
[0092] The processing or the flow shown in drawing 9 generates medium DS from a data model, and, subsequently is related with the operation gestalt which generates a resource code file from medium DS, i.e., another operation gestalt of the processings 607 and 608 of drawing 6 . The flow 900 of this operation gestalt is the same as the flow of drawing 8 except for processings 804, 810, and 816 generating the resource code of specific language, and processings 904, 910, and 916 not generating a resource code, and generating the part of medium DS. Therefore, in case processings 904, 910, and 916 generate the information about the information on a data model, processings 804, 810, and 816 and same processing are performed. However, it may be used for creating the resource code file of resource code language which is comprehensive as for the generated information and is behind different.
[0093] If the DS which should be analyzed to a data model is judged not to exist any longer, medium DS is completed, and a flow will branch and will progress to the creation processing 920. In creation processing, a resource code file is created from medium DS. Since DS is comprehensive description of a resource code file, in processing 920, medium DS is translated into a specific linguistic code file from comprehensive description.
[0094] Drawing 10 is the example of the web page resource file manufactured by the web page developer, i.e., an ASP+ file. A file is the subject of the processing 600 for generating a resource code file. The ASP+ page shown in drawing 10 has an directive line on a line 1, and has a server side control declaration block on a server side script block and lines 16-21 in lines 3-13. Using what kind of type of resource code language, an directive line is used in processing 600, in order to judge whether it is giving description general to it (otherwise, it becoming no code of a resource code file). All the codes written in code declaration block "<script runat=server> </script>" are notionally treated as page member declaration (a variable, a property, method), and are directly inserted in the resource file of a generation file as shown in drawing 11 .
[0095] Drawing 11 is an example of the resource code file created by the processing which creates a resource code using the ASP+ file shown in drawing 10 . The page class which is going to create the 1st line of drawing 11 by the resource code file shows that it has succeeded from System.ASP.WebForms.Page, when compiled. Since the taking over from this class offers many control functions of a page class, it is an important step. As an initial state, the generated resource file is the subclass of a "System.ASP.WebForms.Page" base class. A developer can specify an alternative class as arbitration using the "Inherits" attribute given to the directive line.
[0096] Lines 3 and 4 show the adjustable declaration created while passing along the data model by the 1st pass.
[0097] The information shown in the line 6 at the time of the 2nd pass is created, new control object "DataList" is created, and it is named MyData. In a certain operation gestalt, all the codes currently written in in ASP+ file declaration block "<script runat=server> </script>" are treated as page member declaration (a variable, a property, method), and are directly inserted in the resource file of the above generation files. Thus, the line of a code 8-13 was directly inserted as a literal text. the information relevant to a literal text is not traversed during much pass during creation of a data model 606 -- as -- essential -- the upper part of a data model -- or it separated and was stored by the string or the array (drawing 8). It is judged that this information is directly inserted in the creation time of a data model, and it can be stored as an array which has a reference pointer in a data model.
[0098] One section of a code was written in between the 2nd pass concerning [lines 15-39] control

object information construction. Lines 15-20 express the code used in creation of a top-level object. A top-level object is a container type object of a whole page like the page object 314 as [shown in drawing 3]. Since the ASP+ page by which this control object was created has this type of control object, it is built comparatively independently from the substantial code shown in drawing 10 .

[0099] The lines 22-27 of drawing 11 express the code used for creation of the child control object named MyList. As shown in the line 16 of drawing 10 , table list control was defined and identifier "MyList" was able to be given. The information from the line 16 of drawing 10 is translated into the lines 22-27 shown in drawing 11 in step 810 (drawing 8). essential -- processing 810 -- setting -- the code part of the line 16 of drawing 10 -- analyzing syntax -- if -- a "tablelist" type -- it is recognized as control being created. If recognized, the information on the lines 22-27 shown in drawing 11 will insert a suitable variable like "MyList" for generating the code which searched from the table or was shown in lines 22-27, and will be generated using well-known count. Since the system is designed so that control of this type may be recognized, it can generate the code shown in drawing 11 . If generated, the code will also once write processing 810 in a file.

[0100] Similarly, the lines 29-34 of drawing 11 express a "template" type object in another control object and this case. They are also container control and a template is a special object at the point of being generated actually at the execution time. Anyway, a code required for generation of the control object of a template is generated in step 810 (following the test 812 which judges whether DS still exists in a model). As mentioned above, the template control object should be generated, the code shown in drawing 11 is generated and recognition of being written in the file is made. As for an important thing, the code of a template needs the information relevant to child control so that clearly [reference / to "control3" of the lines 32 and 33 of drawing 11]. Therefore, when the data model was created by 606 (drawing 6), the index which determines this information or this information was stored with template control information. Moreover, the code is beforehand decided to the template control object so that directly [generation of a code].

[0101] Child control is created in a template using the code of the line 19 of drawing 10 , and the translated code corresponding to it in the lines 36-39 of drawing 11 . The line 19 of drawing 10 calls control of the "label" type which has identifier "MyLabel". Like table list control and template control, label control is control of the well-known type which processing 810 recognizes, can generate a code suitable subsequently and can write it in a file as shows it to the lines 36-39 of drawing 11 .

[0102] At the time of the pass of the last of a data model, lines 41-51 were created and generated, and were written in the resource code file. The line of the code in lines 41-51 expresses the rendering method called in order to carry out the rendering of the HTML code which becomes a part of response to a client. A rendering code is generated based on recognition that such a code in step 816 should be generated. Once it is recognized, a code will only be judged from a retrieval table, or will be generated if needed.

[0103] The code shown in drawing 11 expresses the server side resultant code from a specific dynamic web page file, i.e., the file shown in drawing 10 . Completion of the file shown in drawing 11 may compile a file, as the processing 610 shown in drawing 6 was described above. Compile serves as a class which can be used for generating control object control of a dynamic web page contents file. A class is stored in a cache or other memory, and in order to illustrate the object for a page, it can be used the desired number of times. It may be carried out by "in-memory one" so that compile of a result class and neither of a par cis- wardrobe may need access to a disk, and this requires time amount rather than it generally carries out by memory.

[0104] All processings required for the setup of a server side page, starting, and activation are encapsulated in the page class compiled dynamically. Consequently, additional configuration/syntax analysis of a file are not required between page setups. Furthermore, .aspx" from the first or an ASP+ file is not treated again, and "execution-time hosted environment, for example, an ASP script engine," is not required between page executive operation.

[0105] The operation gestalt of this invention in this description is performed as a logic step of one or more computer systems. Logic processing of this invention is performed as a machine module with

which it interconnected within the computer system beyond (2) 1 ** as a processor execute step sequence by which executive operation is carried out by the computer system beyond (1) 1 **. Activation is the problem of selection and it depends for it on the performance requirements of the computer system which performs this invention. Therefore, the logic processing which constitutes the operation gestalt of this invention of a publication on these descriptions can be variously expressed as processing, a step, an object, or a module.

[0106] A description, an above-mentioned example, and above-mentioned data offer the structure of the operation gestalt of this invention, and perfect explanation of an activity. Since this invention can be carried out with many gestalten, without deviating from the pneuma and the range of this invention, this invention is in an attached claim.

[0107]

[Effect of the Invention]

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

CLAIMS

[Claim(s)]
[Claim 1] In the server computer system which has memory in memory In order to create the server side object for carrying out the rendering of the web page contents which are carried by the client side computer system and displayed as a web page on said client computer system dynamically The process which receives the request which is the approach of creating the class used according to said server computer system, and specifies a dynamic web page contents file from said client, The process which generates the resource code file containing the resource code showing the control object which processed said dynamic web page contents file, and was declared in said web page contents file, How to compile said resource code file and create a class including the process which generates the class as which 1 set of hierarchy objects may be illustrated for creation of the web page authoring language which generates the web page to display.
[Claim 2] The approach said dynamic web page contents file creates the class according to claim 1 which is the server side declaration data storage section.
[Claim 3] Said class is the approach of creating the class according to claim 1 which can be used in order to illustrate an object by the response to another request which is stored in the cache memory of said server computer system, and specifies said dynamic web page contents file.
[Claim 4] Said class is the approach of creating the class according to claim 1 which can be used in order to illustrate an object by the response to another request which is stored in a magnetic storage medium and specifies said dynamic web page contents file.
[Claim 5] Since it stores in the data model which contains two or more data objects associated hierarchical in said a part of dynamic web page contents file in down stream processing of said dynamic web page contents file The process which analyzes the functor of said dynamic web page contents file, The process which generates the resource code about declaration information based on the analysis of said data model at the time of the 1st phase, The process which writes the resource code about said declaration information in said resource code file, The process which generates the resource code about control object information based on the analysis of said data model at the time of the 2nd phase, How to create a class including the process which writes the resource code about said control object information in said resource code file at the time of the 2nd phase according to claim 1.
[Claim 6] Said approach is the approach of creating a class including the process which generates the resource code about rendering information based on the analysis of said data model at the time of the 3rd phase, and the process which writes the resource code about said rendering information in said resource code file at the time of the 3rd phase according to claim 5.
[Claim 7] How to create the class according to claim 6 which said 2nd phase will produce if said 1st phase is completed substantially, and said 3rd phase will produce if said 2nd phase is completed substantially.
[Claim 8] How to create the class according to claim 6 which said 1st phase, said 2nd phase, and said 3rd phase produce simultaneously.
[Claim 9] It is the approach of creating the class according to claim 1 which includes further the process

which judges whether the class about said received request is compiled in front of down stream processing of said dynamic web page contents file, and it is stored in memory, and the process of skipping said down stream processing when said class is compiled and it is stored in memory, and performing down stream processing when that is not right.
[Claim 10] Shape is taken by the subcarrier by the computer system which has memory. And in order to create the server side object for carrying out the rendering of the web page contents which are carried by the client side computer system and displayed as a web page on said client computer system dynamically It is the computer data signal which is coding the computer program which carries out executive operation of the computer process which creates in memory the class used according to said server computer system. The process which receives the request said whose computer process specifies a dynamic web page contents file from said client, The process which processes said dynamic web page contents file in order to generate the resource code file containing the resource code showing the control object declared in said web page contents file, A computer data signal including the process which compiles said resource code file in order to generate the web page authoring language which generates the web page to display and to generate the class as which 1 set of hierarchy objects may be illustrated.
[Claim 11] Reading [computer system / which has memory] is possible. And a client side computer system progresses and it is displayed as a web page on said client computer system. In order to create the server side object for carrying out the rendering of the web page contents dynamically It is the computer program storage which is coding the computer program which carries out executive operation of the computer process which creates in memory the class used according to said server computer system. The process at which said computer process receives the request as which a dynamic web page contents file is specified from said client, The process which processes said dynamic web page contents file in order to generate the resource code file containing the resource code showing the control object declared in said web page contents file, A computer program storage including the process which compiles said resource code file in order to generate the web page authoring language which generates the web page to display and to generate the class as which 1 set of hierarchy objects may be illustrated.
[Claim 12] In the server computer system which has memory, it has the web page contents by which the rendering was carried out dynamically. It is the approach of creating two or more web page responses which are carried by one or more client side computer systems, and are displayed as a web page on said client computer system. The process which receives the request which identifies a dynamic web page contents file from said client computer system for said web page, The process which creates the data model for storing the element of said dynamic web page contents file, The process which generates the resource code file about said dynamic web page contents file based on assessment of said data model, They are the process which creates the class which compiled said resource code file and was compiled by memory, and the process which returns a class reference to said server computer system. By this The process which generates web page contents dynamically by enabling said server computer system to illustrate a server side processing object from the class concerned, The process which carries out the rendering of said dynamic web page contents to the web page response carried to said client computer system, The process which leads said web page response to said demanded client computer system, The process which receives the 2nd request which identifies a dynamic web page contents file for said web page, They are the process which judges whether the class by which the dynamic web page contents file concerned was compiled exists in memory, and the process which returns a class reference to said server computer system. By this The process which generates web page contents dynamically by enabling said server computer system to illustrate a server side processing object from the class concerned, How to create a web page response including the process which carries out the rendering of said dynamic web page contents to the 2nd web page response, and the process which leads said 2nd web page response to said demanded client computer system.
[Claim 13] Reading [computer system / which has memory] is possible. And have the web page contents by which the rendering was carried out dynamically, and one or more client side computer systems progress. Are displayed as a web page on said client computer system. It is the computer program storage which codes the computer program which carries out executive operation of the

computer process which creates two or more web page responses. The process which receives a request said whose computer program identifies a dynamic web page contents file from said client computer system for said web page, The process which creates the data model for storing the element of said dynamic web page contents file, The process which generates the resource code file about said dynamic web page contents file based on assessment of said data model, They are the process which creates the class which compiled said resource code file and was compiled by memory, and the process which returns a class reference to said server computer system. By this The process which generates web page contents dynamically by enabling said server computer system to illustrate a server side processing object from the class concerned, The process which carries out the rendering of said dynamic web page contents to the web page response carried to said client computer system, The process which leads said web page response to said demanded client side computer system, The process which receives the 2nd request which identifies a dynamic web page contents file for said web page, They are the process which judges whether the class by which the dynamic web page contents file concerned was compiled exists in memory, and the process which returns a class reference to said server computer system. By this The process which generates web page contents dynamically by enabling said server computer system to illustrate a server side processing object from the class concerned, A computer program record medium including the process which carries out the rendering of said dynamic web page contents to the 2nd web page response, and the process which leads said 2nd web page response to said demanded client computer system.

[Claim 14] Shape is taken by the subcarrier by the computer system which has memory. And have the web page contents by which the rendering was carried out dynamically, and one or more client side computer systems progress. It is the computer data signal which codes the computer program which carries out executive operation of the computer process which creates two or more web page responses displayed as a web page on said client computer system. Said computer program The process which receives the request which identifies a dynamic web page contents file from said client computer system for said web page, The process which creates the data model for storing the element of said dynamic web page contents file, The process which generates the resource code file about said dynamic web page contents file based on assessment of said data model, They are the process which creates the class which compiled said resource code file and was compiled by memory, and the process which returns a class reference to said server computer system. By this The process which generates web page contents dynamically by enabling said server computer system to illustrate a server side processing object from the class concerned, The process which carries out the rendering of said dynamic web page contents to the web page response carried to said client computer system, The process which leads said web page response to said demanded client computer system, The process which receives the 2nd request which identifies a dynamic web page contents file for said web page, They are the process which judges whether the class by which the dynamic web page contents file concerned was compiled exists in memory, and the process which returns a class reference to said server computer system. By this The process which generates web page contents dynamically by enabling said server computer system to illustrate a server side processing object from the class concerned, A computer data signal including the process which carries out the rendering of said dynamic web page contents to the 2nd web page response, and the process which leads said 2nd web page response to said demanded client computer system.

[Claim 15] In order to create the server side object for carrying out the rendering of the authoring language element which is carried by the client side computer system in memory, and is processed on said client computer system dynamically It is the computer program product which codes the computer program which carries out executive operation of the computer process which creates the class used according to a server side computer system by the computer system. The process which receives the request said whose computer program identifies a dynamic web page resource from said client computer system for said resource, The process which processes said resource in order to generate the resource code file about said resource, A computer program product including the process which compiles said resource code file, creates the compiled class in memory, and enables instantiation of the object of said

compiled class.
[Claim 16] It is a computer program product including the process which creates the DS related are the computer program product which codes the computer program which carries out executive operation of the computer process which creates a class in memory by the computer system, and hierarchical [that it is forms the process which analyzes the syntax of said resource in order that said data model creation processing may divide said resource into a logical element and may identify the relation between said logical elements, and a hierarchy data model / plurality], and the process which store the part of said resource in said data configuration according to claim 15.
[Claim 17] Are the computer program product which codes the computer program which carries out executive operation of the computer process which creates a class in memory by the computer system, and it sets to said down stream processing. The process which performs the 1st analysis of said resource in order to generate the resource code relevant to adjustable declaration information, The process which performs the 3rd analysis of said resource in order to generate the resource code relevant to the process and rendering information that the 2nd analysis of said resource is performed in order to generate the resource code relevant to control object information, A computer program product including the process which stores said resource code in said resource code file according to claim 15.
[Claim 18] The computer program product according to claim 16 which is a computer program product which codes the computer program which carries out executive operation of the computer process which creates a class in memory by the computer system and by which said resource code is generated from said medium DS in said resource code generation down stream processing, including further the process which generates medium DS.
[Claim 19] Are the computer program product which codes the computer program which carries out executive operation of the computer process which creates a class in memory by the computer system, and it sets to generation down stream processing of said medium DS. The process which performs the 1st analysis of said resource in order to generate the medium DS element relevant to adjustable declaration information, The process which performs the 3rd analysis of said resource in order to generate the medium DS element relevant to the process and rendering information that the 2nd analysis of said resource is performed in order to generate the medium DS element relevant to control object information, A computer program product including the process which generates a resource code from said medium DS element according to claim 18.
[Claim 20] It is the computer program product according to claim 20 which is a computer program product which codes the computer program which carries out executive operation of the computer process which creates a class in memory by the computer system, and is characterized by for said medium DS being the comprehensive description which may be translated into two or more resource code language files, and at least one resource code file differing from another resource code language file.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

MEANS

[Means for Solving the Problem] A resource code file is compiled by the class in which executive operation is possible about the code generation approach for this invention to create intermediate language or a resource code file from a server side resource, and equipment. Quick generation of the web page control object which performs the server side function containing the rendering of a client response by the class in which executive operation is possible is attained. In an operation gestalt with this invention, a code generation scheme can create the control object connected within the hierarchy, in order to treat setting out of the attribute to event processing and a specific object. Furthermore, the code generation approach can also connect the object declared using the template again.
[0012] This invention relates to the creation approach of the class in server computer system memory more preferably. In order that a class may create the server side object which carries out the rendering of the web page contents dynamically, it is used according to a server computer system, and web page contents are sent to a client side computer system, and are displayed as a web page on a client computer system. In processing, a server computer system receives the request for a web page from a client computer system, and this request identifies a dynamic web page contents file. A server computer creates the data model for storing the element of a dynamic web page contents file, evaluates a data model, and generates the resource code file about a dynamic web page contents file based on assessment of a data model. If a resource code is created, a resource code file will be compiled and will create the compiled class in memory. Generally, a server computer system is enabled to return "refer to the class" to a server computer system, and to use a class for it, and it ends a process.
[0013] According to other desirable operation gestalten, an approach stores a class in the cache memory on a server computer system. If stored in cache memory, many server side page objects will be illustrated from one compiled class, and a resource from the first will not once be used again. It judges whether a server computer system has in memory the class by which the dynamic web page contents file was compiled whenever it received the request to a web page. It is created when the demanded class does not exist in memory. If there is a class, a server computer system will illustrate a server side processing object from the class, in order to generate web page contents dynamically. Next, the rendering of the web page contents is carried out, and they are sent to a client computer system.
[0014] According to still more nearly another operation gestalt of this invention, the approach step which evaluates a data model is accompanied by the repetitive traverse of a data model with two or more pass. A resource code is generated at the time of each pass, and it is written in a resource code file based on assessment of the data model at the time of the pass. A data model is built using the DS connected hierarchical.
[0015] According to a computer system, the operation gestalt of the computer program product by this invention can be read, and contains the computer program storage which codes the computer program which carries out executive operation of the computer process which creates the class compiled by memory on the server computer. The compiled class is used for illustrating a server side processing object in order to carry out the rendering of the response corresponding to the demanded web page which is displayed on a client computer system. Shape is taken by the subcarrier by the computer system

and another operation gestalt of the computer program product by this invention includes the computer data signal which codes the computer program for creating the class compiled by the server.
[0016]
[Embodiment of the Invention] The gestalt of operation of this invention is related with the approach of generating the class compiled by the memory for the specific web page defined by the dynamic web page contents resource or the file. Creation of the compiled class is accompanied by creation of a resource code file from a web page file. Next, a class compiles a resource code file. When the compiled class exists in memory, it may be illustrated in order that a page object may carry out the rendering of the response returned to the client to display. Generally, a page object is accompanied by the server side control object for processing of the client side user interface element displayed on a web page, and generation. Furthermore, the hierarchy of a server side control object may be declared to the web page file which collaborates eventually and generates an authoring language code a result like HTML for the display of these objects of the web page on a client.
[0017] Drawing 1 shows the web server which generates dynamically the web page contents displayed on the client in an operation gestalt with this invention. A client 100 carries out executive operation of the browser 102 which displays a web page 104 on the indicating equipment of a client 100. A client 100 includes the client computer system which has a display like a video monitor (not shown). The "INTERNET EXPLORER" browser currently sold by Microsoft Corp. is an example of the browser 102 in an operation gestalt with this invention. As an example of other browsers, it is "NETSCAPE NAVIGATOR". It reaches. Although there is "MOSAIC" etc., it does not restrict to this. The text box control 106 and two carbon button control 108 and 110 are included in the illustrated web page 104. A browser 102 receives the HTML code from a web server 116 by the HTTP response 112, and displays the web page described by the HTML code. Although HTML is explained with reference to a certain operation gestalt, especially, it is not restricted and it is thought that other authoring languages containing SGML (Standard Generalized Markup Language) and XML (eXtensible Markup Language) are within the limits of this invention.
[0018] The communication link with a client 100 and a web server 116 can be performed using the sequence of the HTTP request 114 and the HTTP response 112. Although HTTP is explained with reference to a certain operation gestalt, it is not restricted especially and it is thought that other transport protocols including S-HTTP are within the limits of this invention. In a web server 116, the HTTP pipeline module 118 analyzes reception and URL for the HTTP request 114, and calls the suitable handler 120 which processes a request. The web server 116 is equipped with two or more handlers 120 treating the resource of a different type in the operation gestalt with this invention.
[0019] For example, when URL specifies a static contents file 122 like an HTML file, a handler 120 accesses the static contents file 122, and sends the static contents file 122 to a client 100 by the HTTP response 112 through the HTTP pipeline 118. Or in an operation gestalt with this invention, when it specifies a dynamic contents resource or a file 124, a handler 120 accesses the dynamic contents file 124, processes the contents of the dynamic contents file 124, and generates the HTML code the result for web page 104. [as / whose URL is an ASP+ (Active Server Page+) page] Generally, a dynamic contents resource like a file 124 is the server side declaration data storage section which can use the authoring language which describes the web page which should be displayed on a client for generating dynamically. And the HTML code for web pages passes along the HTTP pipeline 118, and is sent to a client 100 by the HTTP response 112.
[0020] During this processing, the handler 120 was beforehand developed again, in order to simplify a development effort, or it can access the library of the 3rd person code. One of such the libraries is the server side class control library 126, and a handler 120 can illustrate from here the server side control object which generates HTML data as a result of processing a user interface element and displaying on a web page. in an operation gestalt with this invention, one or more server side control objects are visible on the web page described by the dynamic contents file 124 -- it hides-like and maps to one or more user interface elements.
[0021] A handler 120 accesses one or more non-user interface server components 130 which carry out

executive operation on a web server 116 or another accessible web server again. A non-user interface server component 130 like stock price retrieval application or a database component is referred to in the dynamic contents file 124 processed by the handler 120, or is related with it. The non-user interface server component 130 can process the event started by the server side control object declared by the dynamic contents file 124. Consequently, the processing offered by the server side control object can simplify programming of the non-user interface server component 130 by encapsulating processing and generation of a web page of a user interface element, and, thereby, the developer of the non-user interface server component 130 can be concentrated on development of the function of the application proper instead of a user interface problem.

[0022] Drawing 2 shows the flow chart of processing of the client side user interface element using the server side control object in an operation gestalt with this invention, and generation processing. In processing 200, a client transmits a HTTP request to a server. A HTTP request contains URL which specifies resources, such as an ASP+ page. In processing 202, a server calls the suitable handler which receives a HTTP request and processes the specified resource. Reading appearance of the ASP+ page is carried out in processing 203. Processing 204 generates a server side control object hierarchy based on the content of the specified dynamic contents file (for example, ASP+ page).

[0023] In processing 206, a control object hierarchy's server side control object performs postback event handling, postback data handling, status management, and one or more processings in data coupling. In processing 208, in order that each server side control object in a hierarchy may generate data like the HTML code for the display by the web page of a client side user interface element (or rendering), it is called. Although the vocabulary "a rendering" may mean the processing which displays graphics on a user interface, in this description, the vocabulary "a rendering" also means generation processing of the authoring language data which may be interpreted by client application like the browser for a display and a client side function. More detailed explanation of processing 206 and the rendering processing 208 is given in connection with drawing 6 . The call of the render method in each control object is performed using a tree traversal sequence. That is, the call of the render method of a page object becomes the repetitive traverse covering the suitable server side control object in a hierarchy.

[0024] Or actual creation of each server side control object may delay until a server side control object is accessed in processings 206 or 208 (handling of a postback input, loading of a condition, rendering of a control object to the HTML code, etc.). When a server side control object is not accessed for a given request, server processing is optimized by delaying creation of a control object and eliminating unnecessary control object creation processing.

[0025] In processing 210, the HTML code is transmitted to a client by the HTTP response. In processing 214, a client receives the HTML code relevant to the new web page which should be displayed. In processing 216, a client system displays the user interface element of a new page according to the HTML code received from the HTTP response. A server side control object hierarchy is ended in processing 212. The server side control object in a hierarchy answers the HTTP request which refers to the associated ASP+ page, is created, and after the rendering of authoring language data (for example, HTML data) finishes, it is destroyed. Or processing 212 may be performed before processing 210 after processing 208.

[0026] Drawing 3 shows an example of the module in the web server used in an operation gestalt with this invention. A web server 300 receives the HTTP request 302 to the HTTP pipeline 304. The HTTP pipeline 304 may also contain various modules, such as logging of web page statistics, user collating, a right of user access, and a module for the formation of an output cache of a web page. Eventually, each input HTTP request 302 received by the web server 300 is processed by the specific instance of an interface handler (Interface Handler) (it illustrates as a handler 306), for example, an IHTTP handler class. A handler 306 analyzes a URL request and calls a suitable handler factory (for example, page factory module 308).

[0027] In drawing 3 , the page factory 308 related with the ASP+ page 310 is called, and the instantiation and configuration of an object from the ASP+ page 310 are handled. Although the ASP+ page 310 may be recognized or referred to by URL of a meaning and other prefixes can be used for it by

it, it is further discriminable with the prefix ".aspx", for example. If the request to specific ".aspx" resource is first received by the page factory module 308, the page factory module 308 will search a file system, and will obtain a suitable resource or a suitable file (for example, .aspx page 310). This file may also contain the text (for example, authoring language data) which may be behind interpreted or accessed by the server which processes a request, or the data (for example, cutting tool code data or coded data) in another format. When a physical file exists, the page factory module 308 reads a file to an aperture, and reads the file to memory. Or although the demanded file exists, when beforehand loaded to memory, a resource may not need to be loaded to memory necessarily so that it may state to a detail below. When the demanded aspx file is not found, the page factory module 308 returns the suitable error message "a file is not found" by transmitting for example, HTTP"404" message to a client.
[0028] After reading the ASP+ page 310 to memory, the page factory module 308 processes a file content, and builds the data models (for example, a list [of script blocks], directive, static text area, and hierarchy server side control object, a server side control property, etc.) of a page. A data model is used for generating the source code file of a new object class like the COM+ (Component Object Model+) class to which the class of the page base which is the code which defines the structure, property, and function of a page object is made to extend. In an operation gestalt with this invention, a source list is dynamically compiled by intermediate language. It is behind compiled by the instructions (for example, X86, Alpha, etc.) of a proper on a plat form timely (Just-In-Time). Intermediate language is COM+ IL. A code and Java A cutting tool code and Modula 3 A code and SmallTalk A general purpose or custom-made orientation linguistic code of a code, the Visual Basic code, etc. may also be included. In another operation gestalt, intermediate-language processing is excluded and the instruction of a proper is directly generated from a source list or a source file (for example, ASP+ resource 310). The control class library 312 may be accessed with the page factory module 308, in order to obtain the server side control class which is used for a control object hierarchy's generation and which was defined beforehand.
[0029] The page factory module 308 creates a page object from the compiled class. The page object 314 is a server side control object equivalent to the web page 104 of drawing 1 . The page object 314 and its child object (for example, the text box object 318, the carbon button object 320, and another carbon button object 322) are the control object hierarchy's 316 examples. The example of other control objects can be considered according to this invention, like the custom-made control object, is not restricted especially and also contains the object (after-mentioned) corresponding to the HTML control shown in a table 1. The page object 314 corresponds to the web page 104 of drawing 1 . The text box object 318 corresponds to the text box 106 of drawing 1 . Similarly, the carbon button object 320 corresponds to the additional carbon button 108 of drawing 1 , and the carbon button object 322 corresponds to the deletion carbon button 110 of drawing 1 . The page object 314 relates to other control objects and hierarchy targets on a server. In a certain operation gestalt, a page object is a container object which contains the child object hierarchical. The hierarchical relationship of other gestalten, such as a dependency, can be used with another operation gestalt. In the more complicated control object hierarchy who has the child object of a large number level, one child object may be a container object of other child objects.
[0030] It sets in the above-mentioned operation gestalt, the control object hierarchy's 316 control object is created and performed on a server 300, and each server side control object corresponds to the corresponding user interface element and corresponding logic target on a client. A server side control object collaborates, handles the postback input from the HTTP request 302 again, manages the condition of a server side control object, performs data coupling with a server side database, and generates the authoring language data (for example, the HTML code) used for the display of a web page as a result of a client. Result authoring language data are generated from the server side control object hierarchy 316 (namely, rendering), and are transmitted to a client by the HTTP response 324. For example, in addition to this, control of an ACTIVEX type or when it is processed by the browser, refer to the HTML configuration which produces client side user interface elements (for example, a control carbon button, a text box, etc.) for the result HTML (or other authoring languages) code.
[0031] By declaration made with the ASP+ resource 310, a server side control object can access one or more non-user interface server components 330 for a dialogue with the non-user interface server

component 330 and a client side user interface element. For example, a postback input can be answered and a server side control object can start a server side event to the non-user interface server component registered into those events. Thus, the non-user interface server component 330 minds a user interface element for a dialogue with a user, and it can perform it, without programming a code required displaying and processing these elements.

[0032] Drawing 4 shows the content of an example of the dynamic contents resource in an operation gestalt with this invention. In the illustrated example, the file 400 includes plain text declaration in a certain dynamic contents file format (for example, ASP+). Each declaration gives an instruction to the page factory module 308 which reads a file 400, creates a class, and calls the suitable server side control object which, in addition to this, carries out the rendering of the HTML code or the authoring language which transmits to a client by the HTTP response eventually.

[0033] the 1st line of a file 400 -- the following formats -- setting -- a delimiter -- :<%@ directive which contains a directive between "<%@" and " %>" [attribute=value] %> -- it is here, and especially directive is not restricted and may also contain "page", "cache", or "import". In order to determine buffering semantics, the requirements for a session condition, an error handling scheme, scree PUTINGU language, transaction semantics, and a property like an import directive, a directive is used with the page factory module 308, when processing a dynamic contents file. A directive may exist anywhere in a page file.

[0034] <html> of the 2nd line is a standard HTML initiation tag written in a resource code file as a literal so that additional processing may not be performed in the information for carrying out the rendering of the result HTML code except a direct "write-in" instruction. In HTML functor, <html> shows the beginning of an HTML file and has become the termination tag </html> of the line 21 this [whose] is also a literal, and a pair.

[0035] A code declaration block exists in the lines 3-10 of a file 400. Generally, a code declaration block defines the method by which executive operation is carried out on a page object, a control object member variable, and a server. In the following formats : <script runat = "server" [language = "language"] and [src = "externalfile"] > ..... </script> Here, language and a src parameter are arbitrary. In an operation gestalt with this invention, a code declaration block is defined using the <script> tag including the "runat" attribute which has the value set as the "server." Since the syntax of an inner code is specified, a "language" attribute may be used for arbitration. Although default language can express the language configuration of a whole page, a developer is Jscript by the "language" attribute of a code declaration block. And different language within the same (Practical Extraction and Report Language) web page activation of PERL etc. can be used. The <script> tag can specify a "src" file as arbitration again, and a "src" file is a file of the exterior where a code is inserted in the dynamic contents resource for processing by the page compiler from there. Although the syntax currently indicated is used in this operation gestalt, with another operation gestalt, he should understand that different syntax within the limits of this invention can be used.

[0036] In drawing 4 , two subroutines, AddButton#Click, and DeleteButton#Click are declared in the Visual Basic format into the code declaration block. It will be called, if any subroutine takes two input parameters, "Source", and "E" and a client side click event is detected by the carbon button of a response. In an AddButton#Click subroutine, the text in a user name text box is connected with word "Add", and is loaded to the text data member of a message. In a DeleteButton#Click subroutine, the text in a user name text box is connected with word "Delete", and is loaded to the text data member of a message. Although not shown in drawing 4 , the member variable of a server side control object may be declared with the code declaration block of a file 400. For example, if Visual Basic syntax is used, keyword"DIM" will declare the data variable of a server side control object.

[0037] "A code rendering block" (not shown) may be included in a dynamic contents resource. A code rendering block can contain the code of the amount of the arbitration by which executive operation is carried out by page rendering time amount. In an operation gestalt with this invention, executive operation of the code rendering block is carried out by one "rendering" method by which executive operation is carried out at the time of a page rendering. Executive operation of other parts of a code may

be carried out by the rendering method. A code rendering block fills the following formats (other formats are considered in another operation gestalt).
[0038] <% InlineCode %> "InlineCode" includes the independent language mold code block or flows-of-control block which carries out executive operation on a server at the time of a page rendering here.
[0039] in-line one -- an expression -- again -- being as follows -- instantiation ---like -- syntax -- using -- an expression -- a rendering -- a block -- a delimiter -- " -- < -- % -- @ -- " -- " -- % -- > -- " -- between -- it can use .
[0040] <%= InlineExpression %> Here, the expression included in the "InlineExpression" block is eventually included by the call to "Response.Write (InlineExpression)" of the page object which writes the value from "InlineExpression" in the holder of the suitable location in declaration. For example, "InlineExpression" may be contained in a file 400 as follows.
[0041] <font size = "<%=x%>" > Hi <%=Name%>, you are <%=Age%>! This outputs a greeting and the description about a certain man's age with the font stored in value"x." The man's identifier and age are defined as a string in a code declaration block (not shown). The rendering of the result HTML code is carried out by the server, and it is transmitted to a client by the HTTP response so that the value of "InlineExpression" may be included in a suitable location. That is, it is as follows.
[0042] <font size ="12"> Hi Bob and you are 35! In the line 11 of a file 400, <body> is a standard HTML tag for specifying the beginning of the text of a HTML document. The termination tag </body> is shown in the line 20 of FIIRU 400. In an operation gestalt with this invention, both <body> and </body> are literals.
[0043] The initiation tag <form> of a HTML foam block is seen on a line 12 in this section of the file 400 of drawing 4 . The termination tag </form> of a foam block is seen on the line 19 of HTML file 400. Since a given identifier is related with a foam block, parameter "id" of arbitration can also be included in a HTML control tag, and it enables this to contain many foam blocks in one HTML file.
[0044] The server side label identified by the "message" is declared in the line 18 of a file 400. A "message" label is used in code declared with the lines 5 and 8 of a file 400, in order to display a label on a web page.
[0045] In the foam block, three examples of a HTML control tag corresponding to the user interface elements 106, 108, and 110 of drawing 1 are shown. The 1st user interface element is declared with the line 13 of the file 400 equivalent to a text box. Text literal "User Name" declares the label located in the left-hand side of a text box. The input tag which has type="Text" declares the text box server side control object which considers as the server side control object which carries out the rendering of the text box client side user interface element, and has an identifier called "UserName". The lines 15 and 16 of a file 400 declare the client side user interface element shown as carbon buttons 108 and 110 of drawing 1 , respectively. A "OnServerClick" parameter specifies the suitable subroutine declared with the code declaration block of a file 400. And the server side carbon button control object generated by the response to declaration by the file 400 carries out the rendering of the server side code of the relation which performs the HTML code and carbon button click event of a client side carbon button.
[0046] The text box and carbon button which were declared by the file 400 are the example of HTML server control declaration. In an initial state, the HTML tags in an ASP+ resource are dealt with [no] as literal text contents, and can be accessed in a page developer's programming. However, by specifying using the "runat" attribute which has the value set as "server", the syntax of a HTML tag is analyzed and a page developer can show that it should be treated as accessible server control declaration. Each server side control object can be related with the "id" attribute of the meaning which enables program reference of a corresponding control object at arbitration. The property argument on a server side control object and event association can also be specified using the declaration name / value attribute pair on a tag element (for example, OnServerClick is equal to a "MyButton#Click" pair).
[0047] In an operation gestalt with this invention, the general syntax which declares a HTML control object is as follows.
[0048]
<HTMLTag id = "Optional Name" runat = server> ..... </HTMLTag> Here,

"Optional Name" is the identifier of the meaning of a server side control object. Although the list, the related syntax, and COM+ class of a HTML tag which may be supported are shown in a table 1, other HTML tags can be considered within the limits of this invention.

[0049]

[A table 1]

HTML タグ名	例	COM+ クラス
<a>	 My Link 	AnchorButton
<img>	<img id = "MyImage" runat = server>	Image
		Label
	Some contents	Panel
<form>	<form id = "MyForm" runat = server> </form>	FormControl
<select>	<select id = "MyList" runat = server> <option>One</option> <option>Two</option> <option>Three</option> </select>	DropDownList
<input type = file>	<input id = "MyFile" type = file runat = server>	FileInput
<input type = text>	<input id = "MyTextBox" type = text>	TextBox
<input type = password>	<input id = "MyPassword" type = password>	TextBox
<input type = reset>	<input id = "MyReset" type = reset>	Button
<input type = radio>	<input id = "MyRadioButton" type = radio runat = server>	RadioButton
<input type = checkbox>	<input id = "MyCheck" type = checkbox runat = server>	CheckBox
<input type = hidden>	<input id = "MyHidden" type = hidden runat = server>	HiddenField
<input type = image>	<input type = image src = "foo.jpg" runat = server>	ImageButton
<input type = submit>	<input type = submit runat = server>	Button
<input type = button>	<input type = button runat = server>	Button
<button>	<button id = MyButton runat = server>	Button
<textarea>	<textarea id = "MyText" runat = server> This is some sample text </textarea>	TextArea

[0050] In addition to a standard HTML control tag, an operation gestalt with this invention enables a developer to create the reusable component encapsulated about a program function common to the outside of a HTML tag set. These custom-made server side control objects are specified using the declaration tag in a page file. Custom-made server side control object declaration includes the "runat" attribute which has the value set as "server". In order to carry out possible [of the program reference of a custom-made control object], the "id" attribute of a meaning is specified as arbitration. Furthermore, the declaration name / value which is an attribute pair in a tag element specify the property argument of a server side control object, and event association. An in-line template parameter may also be combined with a server side control object by giving the element of the child who is the prefix-letter train of a suitable "template" to a parent server control object. A format of custom-made server side control object declaration is as follows.

[0051] <servercntrlclassname id="OptionalName" [propertyname="propval"] runat=server/> Here,

"servercntrlclassname" is an accessible control class, "OptionalName" is the identifier of the meaning of a server side control object, and "propval" expresses the property value of the arbitration of a control object.
[0052] An XML tag prefix can be used for offering the briefer notation for specifying a server side control object in a page by using the following formats using another specification statement method.
[0053] <tagprefix:classname id = "OptionalName" runat = server/> Here, in "tagprefix", in relation to a given control name tooth-space library, "classname" expresses the identifier of control in a relation name tooth-space library. "propertyvalue" of arbitration is supported.
[0054] If drawing 5 is referred to, an example of the computer system of the operation gestalt of this invention contains the general purpose computer equipment of the gestalt of the conventional computer system 500 containing the system bus 506 which connects the various system components containing the processor unit 502, a system memory 504, and a system memory 504 to the processor unit 500. System buses 506 may be any of the bus structure of some types containing the peripheral bus and local bus which use a memory bus or a memory controller, and various bus architectures. The system memory contains the memory (ROM) 508 only for playbacks, and random access memory (RAM) 510. The unformatted input / output system 512 (BIOS) containing the basic routine which helps a transfer of the information between the elements within a computer system 500 are stored in ROM508.
[0055] The computer system 500 contains the optical disk drive 518 which performs read-out and the writing of a removable optical disk 519 like the magnetic disk drive 514 and CD ROM which perform further read-out and the writing of a hard disk drive 512 and the removable magnetic disk 516 which perform read-out and the writing of a hard disk, DVD, or other optical media. The hard disk drive 512, the magnetic disk drive 514, and the optical disk drive 518 are connected by the hard disk drive interface 520, the magnetic disk drive interface 522, and the optical disk drive interface 524 system bus 506, respectively. A drive and its medium which can be related computer read offer the instruction of a computer system 500 which can be computer read, DS, a program, and the non-volatile storage section of other data.
[0056] Although the hard disk, the removable magnetic disk 516, and the removable optical disk 519 are used for this description in the above-mentioned environmental example of a publication, the medium type [other] in which data storage is possible and which can be computer read can be used for the above-mentioned example of a system. The medium of other types of these which can be used for the above-mentioned example of operating environment which can be computer read has for example, a magnetic cassette, flash memory card, a digital videodisc, a BERUNUI (Bernoulli) cartridge, random access memory (RAM), the memory (ROM) only for playbacks, etc.
[0057] Many program modules are stored in a hard disk, a magnetic disk 516, an optical disk 519, and ROM508 or RAM510, and these contain the application program 528, other program modules 530, and the program data 532 of 526 or 1 or more operating systems. A user can input a command and information into a computer system 500 with input units, such as a keyboard 534 and a mouse 536, or other pointing equipments. As other input devices, there are a microphone, a joystick, a gamepad, a satellite dish, a scanner, etc., for example. These and other input devices are connected to the processor 502 through the serial port interface 540 connected to the system bus 506 in many cases. However, these input devices may be connected by other interfaces, such as a parallel port, a game port, or Universal Serial Bus (USB), again. The monitor 542 or the indicating equipment of other types is also connected with the system bus 506 through the interface of a video adapter 544 etc. In addition to a monitor 542, typically, a computer system contains other circumference output units (not shown), such as a loudspeaker and a printer.
[0058] A computer system 500 can operate in the environment using the logical connection to one or more remote computers like a remote computer 546 connected by network. a remote computer 546 -- a computer system, a server, a router, Network PC, and a pier (peer) -- it may be equipment or other common network nodes, and there are many elements typically mentioned above in connection with a computer system 500, or all are included. Network connection contains Local Area Network (LAN) 548 and Wide Area Network (WAN) 550. Such a network environment is not new in office, an enterprise

magnitude computer network, intranet, and the Internet.
[0059] When using by the LAN network environment, a computer system 500 is connected to a local network 548 through a network interface or an adapter 552. When using by the WAN network environment, a computer system 500 includes the modem 554 or other means for establishing the communication link by Wide Area Network 550 like the Internet typically. Built-in or external any is sufficient as a modem 554, and it is connected with the system bus 506 through the serial port interface 540. In the environment connected by network, the program module described in relation to the computer system 500 or its part may be memorized by remote memory storage. The illustrated network connection is an example and can use other means for the communication link establishment between computers.
[0060] In the operation gestalt of this invention, a computer 500 expresses a web server and CPU502 carries out executive operation of the page factory module on the ASP+ file memorized by at least one of storages 516, 512, 514, 518, and 519 or memory 504. A HTTP response and a request communicate by LAN548 connected to the client computer 546.
[0061] The processing 600 performed with the page factory module 308 is shown in the flow chart of drawing 6 . In drawing 6 , processing 600 is mostly equivalent to generation processing (drawing 2) of the control object hierarchy 204. Processing 600 is changed into the object code class which had the ASP+ page (it is also called an ASP+ file) compiled, and subsequently, it is loaded to memory, in order to illustrate the control objects 314, 318, 320, and 322 (drawing 3).
[0062] If a server receives the request to URL in processing 202 (drawing 2), the page creation processing 600 will start. The syntax-analysis processing 602 analyzes syntax of demanded URL after reception of a request, and it judges which resource is demanded. Including files (.htm, .gif, .jpg, etc.) with a static resource, directory browsing starting, a DAV (digital audio/video) file, and dynamic contents requests (.aspx, .soap, etc.), this invention is constituted so that these may be handled. The HTTP request to which close [which was received by the server side] comes is eventually processed by the specific instance of a handler class, for example, an IHTTPHandler class. The architecture which interprets the URL request to a handler instance, for example, an IHTTPHandler instance, actually by the handler factory, for example, the activity of IHTTPHandler "a factory", and in which a spigot is possible is offered. Close can perform this interpretation easily using application configuration setting out which can map the file extension and the HTTP command of URL by which it comes to a factory class like the IHTTPHandlerFactory class which is to create eventually a suitable handler instance, for example, an IHTTPHandler instance. In syntax-analysis processing, although various resources are discriminable with spigot possible architecture, the following publications focused on the HTTP request which identifies a dynamic web page contents resource, and have focused on the .aspx or ASP+ page or a dynamic web page contents file like a file especially.
[0063] If a actual resource is identified by the syntax-analysis processing 602, in the check processing 603, memory will be checked and it will judge whether a class exists in memory. In order to judge whether the class is compiled or not, in the check processing 603, the class in memory is searched by searching the flag which will be set if a class is compiled, or the identifier. Whichever it makes it, in the check processing 603, the index which shows that a compile class is already compiled and is stored is searched. When a class is in memory, a flow branches on a YES branch and progresses to the instantiation processing 612. Instantiation processing illustrates a control object from the compiled class. In the check processing 603, if it is judged that the class is not compiled, a flow will branch on NO branch and will progress to the fixing processing 604.
[0064] If the check processing 603 judges that a class does not exist in memory, in the fixing processing 604, it will search and discover a specific resource and will read a file to memory. A base class like a "System.ASP WebForms.PageFactory" class offers the handler factory implementation which handles instantiation and configuration of ASP and an ASPX page. If a resource and a dynamic web page contents file physical in this case are not found, a suitable error message is returned.
[0065] Next it is the fixing processing 604, in processing 600, syntax-analysis creation processing 606 is performed and syntax of a resource like an ASP+ file is analyzed here. Syntax-analysis/creation

processing 606 creates a data model from the information collected while reading and analyzing the syntax of an ASP+ file for every declaration. A data model is the DS containing the element relevant to the element of the active content file which can pull out a resource code. Including the structural element with which the data model was referred to by the active content file, these elements are connected so that the structure of a control object may be expressed as a result of an active server page. In an operation gestalt with this invention, a data model is the combination of the object associated by the hierarchical tree structure.

[0066] Each declaration of a web contents file has the predetermined element in which the information stored in the creation approach of a data model and a data model is shown. For example, when declaration is literal text declaration, a data model should just include information by which a text is inserted in a suitable location. However, when declaration shows that a nested control object exists, the data model according to a nest which was declared to the ASP+ file must be created. Intrinsically, as for a data model, as for each object, a child object includes the information about whether it exists or not in each object including one object for every declaration.

[0067] If the syntax of an ASP+ file is analyzed and a data model is created, arbitrary creation processings 607 may be performed. The creation processing 607 creates the medium DS which abstracts a resource code like the resource code file which should analyze a data model, for example, should be created so that it may indicate below. Intrinsically, medium DS is comprehensive DS which describes a code. Medium DS has the DS of the upper level which describes a class. You may be the DS of another level which has the list or array of a class name and a method for every class. Furthermore, it is the DS which has each method itself, a statement, and various elements like declaration. A statement may include items, such as an expression and a data call.

[0068] If the syntax of an ASP+ file is analyzed and a data model (and arbitration medium DS) is created, typically, the creation resource code module 608 will create a required resource code through the analysis of each data model by writing in the various lines of the code of each declaration which exists in an ASP+ file. Generally, this step is accompanied by the well-known declaration in a data model, and direct translation in the code of other information. The hard coding of such a translation is carried out to the application which translates, or each translation may be stored in a retrieval table. Although this kind of retrieval table may be created so that the suitable translation for almost all language may be offered, the compile to COM or the COM+ class for using existing COM or an existing COM+ library becomes easy by the activity of object-oriented resource code language. (For example, there is C++ or Vbasic as an example of resource code language.) Further, when medium DS is created by processing 607, a translation becomes still more direct. In such a case, DS may become very close to a comprehensive resource code language file, and a translation may become the addition of an only suitable lexicon to create the resource code file of specific resource code language, and syntax.

[0069] An ASP+ file declares specific resource code language to a result resource code file. When language is not declared to an ASP+ file, default language, such as Visual Basic, can be used. As a result of being generated from a data model or medium DS, a resource code file is a resource code file with the advanced type which will be intrinsically required to write in when not receiving the benefit of this code generation tool by the case where a programmer uses a server side control object hierarchy.

[0070] Completion of a resource code file creates the class which compiled the resource code file and was compiled in object code or other executive operation possible forms in the compile processing 614. Or a class can be compiled in the language of the made others started on a cutting tool code or a virtual unit. Compile of a resource code file is accompanied by the activity of the compiler constituted so that the resource code language which exists in a resource code file might be compiled. And a class is called by the instantiation processing 616 and creates a control object as a result of the page object for a web page. Although not clearly equipped in a URL request, instantiation processing is un-explicit processing which is a part of URL request of a specific resource.

[0071] A page class's compile of an ASP+ file maintains a class in the usable condition by the future request. therefore, generation processing of the resource code of an ASP+ file is performed once -- sufficient -- a next request can use the compiled class and illustrates a required object if needed. As for

the compiled class, the between cache of the short period of time may be carried out, and/or the compiled class can be stored in a disk. Once an ASP+ file is compiled, it is not actually necessary to touch an ASP+ file from the first again. Of course, when an ASP+ file is corrected, a page class is updated by repeating compile processing. It can judge whether the ASP+ file was updated using a suitable flag or other index mechanisms, therefore renewal of an automatic class is possible. Or after updating ASP+ to a web page file, the charge of removing the class by which the cache was carried out may be left behind to a developer.

[0072] The detail relevant to creation of a data model is shown in drawing 7 . The data model creation processing 700 begins from judging whether in an ASP+ file, there is any declaration by the test processing 702. When there is no declaration like a pure HTML file which should be compiled, a flow branches on NO branch and progresses to termination of processing 714.

[0073] If it is judged that an ASP+ file has at least one declaration by the test processing 702, a flow will branch on a YES branch and will progress to the acquisition processing 704 which obtains the 1st declaration by the ASP+ file. The information from the 1st declaration is stored in DS by the storing processing 706. Not only the actual declaration stored in DS but the information relevant to the 1st declaration of whether whether control of the container type with which declaration has a child object being declared, and declaration declare a directive etc. may be stored in DS. Since a web page is an ASP+ file, it has an directive tag as the 1st declaration typically. a directive -- a tag -- versatility -- a directive -- for example, -- drawing 4 -- a line -- one -- being shown -- as -- a delimiter -- " -- < -- % -- @ -- " -- " -- % -- > -- " -- between -- information -- containing . The delimiter shown in drawing 4 is instantiation-like only, and should understand that other selections are possible. A directive supplies information to the page factory module used for resource code file creation. For example, the directive shown in the line 1 of drawing 4 shows that the default language which should be used is VB (namely, Visual Basic) in this example.

[0074] If the 1st declaration is taken out and it is stored in DS, processing 700 will progress to the test processing 708 which judges whether a file has another declaration. This processing is the same as the above-mentioned processing 702. If there is already no declaration, a flow will branch on NO branch and will progress to the termination step 714. If there is another declaration, a flow will branch on a YES branch, will progress to the acquisition processing 710, and will gain the next declaration.

[0075] It is the ejection of the next declaration, next the storing processing 708 stores the information relevant to the next declaration in different DS related with other DS of a data model. As the 1st declaration was described in the top, the information stored in DS may include not only the literal text of the next declaration but the information derived from the literal text. Therefore, DS may be related with the 1st DS hierarchical if needed.

[0076] Next it is storing of the information on the following DS, a flow branches to the test processing 708 and it judges whether the declaration which should be carried out reading appearance to a resource file, for example, an ASP+ file, at a data model still exists. When it does not exist, a flow branches on NO branch and progresses to termination of processing. however . If the test processing 708 judges that declaration still exists, a flow will branch on a YES branch and will progress to the acquisition processing 710 which gains the next declaration. Next it is the acquisition processing 710, in the storing processing 708, the information relevant to the next declaration is stored as mentioned above. These steps 708, 710, and 712 continue until all declarations are taken out one by one and stored in DS. DS may be associated as a hierarchy connected so that they can identify the sub element of a nested object, a child object, a node, or a parent object as a sub element.

[0077] In creation of the resource code module 608 (drawing 6), the resource code file of specific resource code language is created using this data model. It is important for creation processing of the resource code file from a data model to note being dependent on resource code language. That is, the resource code file written in in specific language has typically the specific requirements for a format, for example, the requirements that all adjustable declarations precede the activity of these variables. Therefore, if the compiler of the language is called, suitable program instruction must be arranged in the suitable location in a resource code file. In order to make and accomplish this activity, an ASP+ file

must be evaluated, and it must judge of what kind of meaning an element or the description is in a file, and a file must be processed, and a resource code file must be generated. Since, as for a resource code file, the thing in beginning of a file typically like adjustable declaration of the element of the 1st lot or declaration is required, the whole data model should be analyzed for this information. Similarly, since there should be control object information in the center of a file, the whole data model should be analyzed for this information at the time of the 2nd next phase of the 1st phase of processing.
[0078] These phases are functional processings which judge a part with the separate code which should be written in a resource code file (or medium DS in processing 607), and a separate part is together put logically here based on a location the result in a resource code file. Therefore, these phases are considered to be "pass" or traverse of one data model, therefore are performed continuously. however -- or these phases may be performed as separate concurrent processing, the result of separate processing is summarized in one resource code, and processing evaluates either of the separate copies of a single data model or a data model. Or the functional processing or the functional phase which judges and writes in a part with a separate code may be performed at the time of a single traverse, and a resource code is selectively written in a part with a separate resource code file, or is written in the separate file which was summarized so that the part might offer the suitable connection between the logical parts of a resource code, or was associated there. Therefore, in this description, although it has indicated substantially that it is carried out in succession by separate analysis one after another, the split of the analysis may be carried out simultaneously substantially, or it may be performed.
[0079] The processing or the flow shown in drawing 8 is related with the operation gestalt 607 which generates a direct resource code file comparatively from a data model, i.e., the abbreviation processing shown in drawing 6 . A variable is declared first and this operation gestalt generates a resource code file like the VB code file to which an object and a method follow variable information. The phase when the resource code creation processing 800 shown in drawing 8 is separate as "pass" of a data model is shown, and three times pass the DS of a data model. A variable and declaration information are looked for in processing of the 1st pass. In processing of the 2nd pass, object creation information is looked for and code rendering information is looked for in the 3rd pass processing 800.
[0080] The 1st pass starts in the processing 802 which traverses a data model about a variable and other declaration information. This step begins from analyzing the 1st DS of the data model created by processing 700. In analysis processing of the 1st DS, it judges whether there is any specific line of the resource code which should be written in the resource code file about declaration.
[0081] Termination of the analysis of the 1st DS generates and writes the resource code about a variable and declaration information in a resource code file in processing 804. A writer object is called and the text given to it is only written in a resource code file. Therefore, the call to a writer object contains both text strings of the syntax of the specific programming language about the parameter, the variable, or declaration information on a resource code file name. Or a text is copied or is written in a resource code file using other well-known means.
[0082] Moreover, the suitable syntax which should be passed to a writer object must be determined in processing 804. Intrinsically, when current DS expresses server side control, it generates the resource code to it in processing 804. Otherwise, in processing 804, it is copied to a resource code file. Intrinsically, processing 804 can perform one of three. In the generation processing 804, you may judge whether it follows [whether the information in an ASP+ file is a literal text, and] first, and should be directly written in a resource code file. This is the case where a HTML tag is inserted in an ASP+ file, and resource code declaration is not generated in this case. Instead, information is only copied to a resource file. In processing 804, you may judge whether 2nd the response of 1 to 1 is required in the suitable syntax form of the translation with a simple text, and specific resource code language. In such processing 804, when a hard code is carried out to a page factory module in search of 1 to 1 response, one-for-one translation is performed, and result information is supplied to a writer object. When one-for-one translation is not made [3rd], in processing 804, a more complicated translation is required, therefore a module or the next retrieval table processing is called, and creation processing of the suitable resource code syntax which should be passed to a writer object is performed. Typically, the translation

of declaration information is direct.
[0083] In processing 804, if the 1st the generation and writing of a resource code of DS are performed, in the test processing 806, it will detect whether the DS which should be analyzed to a data model still exists. As for a flow, in a certain case, the DS which should be analyzed at the time of this 1st pass of a data model still branches on a YES branch, and it traverses to the following DS at the beginning of processing 800 in return and 802. In such a case, the following DS is analyzed as mentioned above, generates resource code syntax, and calls the writer object which writes the syntax code in a resource file.
[0084] In the test processing 806, when it is judged that the DS which should be analyzed to a data model at the time of this 1st pass does not exist any longer, a flow branches on NO branch, progresses to the traverse processing 808, and starts the 2nd pass which passes DS. The traverse processing 808 starts the next phase of writing in the code which writes in an object fabrication and method creation typically. The traverse processing 808 begins from on a data model, and returns to the DS which a data model begins.
[0085] Next, the 1st DS is analyzed in generation and the write-in resource code processing 810, and it judges whether the information about object creation is in the 1st DS. When such object creation information exists in the 1st DS, it is transmitted to the writer object which the syntax of specific programming language is created and generated, and writes in a resource code file. Intrinsically, processing 810 is the same as the above-mentioned processing 804, if DS is analyzed about the information on a particular type and the information is collected from DS, will perform a certain translation and will create the resource code language about the information here. When information is a literal, in processing 810, the information can be directly sent to a resource code file. Similarly, when the information needs an easy translation, the easy translation may be a part of whether a hard code is carried out to a program, and retrieval table type processing. Furthermore, when complicated structure is required, a retrieval table or other modules are called, information is processed, and the suitable syntax resource code for a resource code file is created.
[0086] It is processing 810, next the enquiry step 812 tests whether the DS which should be analyzed to a data model exists. When the DS which should be analyzed for object creation exists in a data model, a flow progresses to the traverse processing 808 which branches on a YES branch and traverses the following DS. Therefore, in processing 808, the DS of the information about object creation is analyzed in return and processing 810 to the following DS. As mentioned above, it is transmitted to a writer object and this object creation information is translated into the resource code line which can be added to a resource code file. Steps 812, 808, and 810 are repeated until all resource codes are written in about object creation.
[0087] If the decision step 812 judges that the DS which should be analyzed for object creation does not exist in a data model any longer, a flow will branch on NO branch, and will progress to processing 814, and it will end the 2nd pass. This 3rd pass starts in the processing 814 of the top of a data model. As processings 808 and 802 were described in the top, this traverse step takes out the 1st DS in a data model first. It is the ejection of the 1st DS in the data model in processing 814, next the DS of code rendering information is analyzed by processing 816. It is transmitted to a writer or code rendering information is translated into the syntax [****] in the programming language made to add to a resource code file. A code rendering method is near the resource code end of file, therefore the 3rd pass is performed to the degree of object creation pass. Or a code rendering method is generated in between by simultaneous thread processing, and is added to a resource end of file.
[0088] Processings 816 may be collected from the above-mentioned processings 810 and 804 or the code rendering information which becomes and is alike and this resource code information has in DS. Furthermore, you may require as mentioned above that information should be inserted directly, should be translated at a rate of 1 to 1, and should generate the syntax of a resource code file using a more complicated module. Next it is processing 816, in the detection processing 818, it judges whether DS remains in the data model.
[0089] In processing 818, if it is detected as the DS which should be analyzed between this 3rd pass still

existing, a flow will branch on a YES branch and will return to 814 for the traverse of the following DS. The following DS is called and the generation and the writing of a resource code about the code rendering information on the following DS are performed in processing 818.
[0090] In ** Li 818, if it judges that the DS which should be analyzed does not exist any longer, a flow will branch on NO branch and will end processing by the processing connection 820.
[0091] Several pass is performed by the above-mentioned explanation to a data model so that clearly. Each declaration of a data model depends this on requiring the specific line of the code put on the various locations of a resource code file. However, in case a resource code file is written in, a writer object can only perform adding to a file continuously and writing in it one by one, and cannot insert a resource code line between the code lines written in beforehand. Therefore, a data model must be traversed once or more, in order to collect the suitable information belonging to the part of the 3rd or the last of the 2nd or the pars intermedia of the 1st or the upper part of a resource code, and a resource code, and a resource code file. It is expected that the number of pass changes with number of the parts of the code which exists to various computer languages.
[0092] The processing or the flow shown in drawing 9 generates medium DS from a data model, and, subsequently is related with the operation gestalt which generates a resource code file from medium DS, i.e., another operation gestalt of the processings 607 and 608 of drawing 6 . The flow 900 of this operation gestalt is the same as the flow of drawing 8 except for processings 804, 810, and 816 generating the resource code of specific language, and processings 904, 910, and 916 not generating a resource code, and generating the part of medium DS. Therefore, in case processings 904, 910, and 916 generate the information about the information on a data model, processings 804, 810, and 816 and same processing are performed. However, it may be used for creating the resource code file of resource code language which is comprehensive as for the generated information and is behind different.
[0093] If the DS which should be analyzed to a data model is judged not to exist any longer, medium DS is completed, and a flow will branch and will progress to the creation processing 920. In creation processing, a resource code file is created from medium DS. Since DS is comprehensive description of a resource code file, in processing 920, medium DS is translated into a specific linguistic code file from comprehensive description.
[0094] Drawing 10 is the example of the web page resource file manufactured by the web page developer, i.e., an ASP+ file. A file is the subject of the processing 600 for generating a resource code file. The ASP+ page shown in drawing 10 has an directive line on a line 1, and has a server side control declaration block on a server side script block and lines 16-21 in lines 3-13. Using what kind of type of resource code language, an directive line is used in processing 600, in order to judge whether it is giving description general to it (otherwise, it becoming no code of a resource code file). All the codes written in code declaration block "<script runat=server> </script>" are notionally treated as page member declaration (a variable, a property, method), and are directly inserted in the resource file of a generation file as shown in drawing 11 .
[0095] Drawing 11 is an example of the resource code file created by the processing which creates a resource code using the ASP+ file shown in drawing 10 . The page class which is going to create the 1st line of drawing 11 by the resource code file shows that it has succeeded from System.ASP.WebForms.Page, when compiled. Since the taking over from this class offers many control functions of a page class, it is an important step. As an initial state, the generated resource file is the subclass of a "System.ASP.WebForms.Page" base class. A developer can specify an alternative class as arbitration using the "Inherits" attribute given to the directive line.
[0096] Lines 3 and 4 show the adjustable declaration created while passing along the data model by the 1st pass.
[0097] The information shown in the line 6 at the time of the 2nd pass is created, new control object "DataList" is created, and it is named MyData. In a certain operation gestalt, all the codes currently written in in ASP+ file declaration block "<script runat=server> </script>" are treated as page member declaration (a variable, a property, method), and are directly inserted in the resource file of the above generation files. Thus, the line of a code 8-13 was directly inserted as a literal text. the information

relevant to a literal text is not traversed during much pass during creation of a data model 606 -- as -- essential -- the upper part of a data model -- or it separated and was stored by the string or the array (drawing 8). It is judged that this information is directly inserted in the creation time of a data model, and it can be stored as an array which has a reference pointer in a data model.

[0098] One section of a code was written in between the 2nd pass concerning [lines 15-39] control object information construction. Lines 15-20 express the code used in creation of a top-level object. A top-level object is a container type object of a whole page like the page object 314 as [shown in drawing 3]. Since the ASP+ page by which this control object was created has this type of control object, it is built comparatively independently from the substantial code shown in drawing 10 .

[0099] The lines 22-27 of drawing 11 express the code used for creation of the child control object named MyList. As shown in the line 16 of drawing 10 , table list control was defined and identifier "MyList" was able to be given. The information from the line 16 of drawing 10 is translated into the lines 22-27 shown in drawing 11 in step 810 (drawing 8). essential -- processing 810 -- setting -- the code part of the line 16 of drawing 10 -- analyzing syntax -- if -- a "tablelist" type -- it is recognized as control being created. If recognized, the information on the lines 22-27 shown in drawing 11 will insert a suitable variable like "MyList" for generating the code which searched from the table or was shown in lines 22-27, and will be generated using well-known count. Since the system is designed so that control of this type may be recognized, it can generate the code shown in drawing 11 . If generated, the code will also once write processing 810 in a file.

[0100] Similarly, the lines 29-34 of drawing 11 express a "template" type object in another control object and this case. They are also container control and a template is a special object at the point of being generated actually at the execution time. Anyway, a code required for generation of the control object of a template is generated in step 810 (following the test 812 which judges whether DS still exists in a model). As mentioned above, the template control object should be generated, the code shown in drawing 11 is generated and recognition of being written in the file is made. As for an important thing, the code of a template needs the information relevant to child control so that clearly [reference / to "control3" of the lines 32 and 33 of drawing 11]. Therefore, when the data model was created by 606 (drawing 6), the index which determines this information or this information was stored with template control information. Moreover, the code is beforehand decided to the template control object so that directly [generation of a code].

[0101] Child control is created in a template using the code of the line 19 of drawing 10 , and the translated code corresponding to it in the lines 36-39 of drawing 11 . The line 19 of drawing 10 calls control of the "label" type which has identifier "MyLabel". Like table list control and template control, label control is control of the well-known type which processing 810 recognizes, can generate a code suitable subsequently and can write it in a file as shows it to the lines 36-39 of drawing 11 .

[0102] At the time of the pass of the last of a data model, lines 41-51 were created and generated, and were written in the resource code file. The line of the code in lines 41-51 expresses the rendering method called in order to carry out the rendering of the HTML code which becomes a part of response to a client. A rendering code is generated based on recognition that such a code in step 816 should be generated. Once it is recognized, a code will only be judged from a retrieval table, or will be generated if needed.

[0103] The code shown in drawing 11 expresses the server side resultant code from a specific dynamic web page file, i.e., the file shown in drawing 10 . Completion of the file shown in drawing 11 may compile a file, as the processing 610 shown in drawing 6 was described above. Compile serves as a class which can be used for generating control object control of a dynamic web page contents file. A class is stored in a cache or other memory, and in order to illustrate the object for a page, it can be used the desired number of times. It may be carried out by "in-memory one" so that compile of a result class and neither of a par cis- wardrobe may need access to a disk, and this requires time amount rather than it generally carries out by memory.

[0104] All processings required for the setup of a server side page, starting, and activation are encapsulated in the page class compiled dynamically. Consequently, additional configuration/syntax

analysis of a file are not required between page setups. Furthermore, .aspx" from the first or an ASP+ file is not treated again, and "execution-time hosted environment, for example, an ASP script engine," is not required between page executive operation.

[0105] The operation gestalt of this invention in this description is performed as a logic step of one or more computer systems. Logic processing of this invention is performed as a machine module with which it interconnected within the computer system beyond (2) 1 ** as a processor execute step sequence by which executive operation is carried out by the computer system beyond (1) 1 **. Activation is the problem of selection and it depends for it on the performance requirements of the computer system which performs this invention. Therefore, the logic processing which constitutes the operation gestalt of this invention of a publication on these descriptions can be variously expressed as processing, a step, an object, or a module.

[0106] A description, an above-mentioned example, and above-mentioned data offer the structure of the operation gestalt of this invention, and perfect explanation of an activity. Since this invention can be carried out with many gestalten, without deviating from the pneuma and the range of this invention, this invention is in an attached claim.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

PRIOR ART

[Description of the Prior Art] A typical web browser receives data from the web server which defines the appearance of a web page and basic actuation which are displayed on a client system. As a typical procedure, a user specifies the uniform resource (resource) locator ("URL") which is the global address of the resource on World Wide Web, and a desired website is accessed. An example of URL is "HYPERLINK "http://www.microsoft.com/ms.htm" http://www.microsoft.com/ms.htm." The 1st part of this example of URL shows the given protocol (for example, "http") used for a communication link. The 2nd part specifies the domain name (for example, "HYPERLINK "http://www.microsoft.com" www.microsoft.com") which shows the location of a resource. The 3rd part specifies the resource in a domain (for example, file called "ms.htm"). This is followed and they are a browser and HYPERLINK "http://www.microsoft.com" www.microsoft.com. The HTTP (hypertext transport protocol) request relevant to the example of URL for taking out the data relevant to the ms.htm file in a domain is generated. The web server which is acting as the host of the www.microsoft.com site returns reception, the demanded web page, or a resource to a client system by the HTTP response, and displays a HTTP request on a browser.
[0003] The "ms.htm" file of the above-mentioned example corresponds with the web page file containing the static HTML (HyperText Markup Language) code. HTML is a plane (plaintext) text authoring language used for the document (for example, web page) creation on World Wide Web. Since it is such, an HTML file is taken out from a web server by the client browser which changes the HTML code into a actual visual image or a voice component, and is displayed as a web page. In a client computer system, this process displays the content of a web page defined as the passed HTML file. A developer specifies the text and list which are displayed on a browser and by which formatting was carried out, form, a table, a hypertext link, an in-line image, voice, and background graphics using HTML. However, an HTML file is a static file which is not supporting dynamic generation of web page contents in essence. Web page contents are the HTML codes returned to a client for a display. Such dynamic processing is related with the server side application which generates the HTML code in advance of transmission as a result of a processing step rather than only transmits a predetermined code to a client browser.
[0004] In order to handle the dialogue between more complicated client-server, the server side application program has been developed so that the dialogue between more complicated client-server which gives dynamic contents, for example, the stock price which is changing, or traffic information may be handled. A server side application program processes a HTTP request, and generates the suitable HTML code for transmission to a client by the HTTP response to a client. For example, a server side application program can process the enquiry string offered by the client side in a HTTP request, or the data from web based form, in order to generate dynamically the transmitting HTML code in the HTTP response to a client side. Intrinsically, server side application can generate the HTML mold file customized based on the information in the request from a client side. In this case, the static HTML file stored on the server does not exist, but an HTML file is dynamically created at the time of activation. The HTML code may be generated using the sequence of one or more text write-in processings to a

memory device by which formatting was carried out as an example of a server side application program. Then, it is transmitted to a client system by the HTTP response, and the obtained text is displayed on a browser there.

[0005] Development of a server side application program is a complicated activity as which it is required it is not only well versed in the usual HTML coding used for a web page design, but that it is well versed in pro GURAMUBE six including one or more programming language (for example, C++, Perl, Visual Basic, or Jscript). However, although a web page architect is graphic designer or an editor in many cases and gives humane sensibility to a regrettable thing, it is inexperienced in programming in many cases. Therefore, it is required for a man with few programming experiences to offer the web page development framework simplified for creating the web page file which can develop the web page interface of server side application and each of its client. Therefore, to offer the development framework to which a developer can create and process a web page dynamically by the minimum programming is desired.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

TECHNICAL PROBLEM

[Problem(s) to be Solved by the Invention] One means which makes min the requirements for a program of dynamic web page generation is an active server page (ASP) framework offered by Microsoft Corp. By the ASP framework, a developer is Visual Basic or Jscript typically. The "ASP" web page file containing a code and the other HTML codes can be created. An ASP file contains the declaration or the tag and VB script, or J script code which performs various functions. It is easier to write in these declarations generally, rather than it writes in a actual programming code.
[0007] During processing, a HTTP request specifies an ASP file as a desired resource, and generates the HTML code after that using an ASP file as a result of the HTTP response to a client side. Furthermore, in order to reduce a given application programming effort, or the ASP file was developed beforehand, refer to a third party's client side library component (for example, client side "ACTIVEX" control) and a database, or other 3rd person applications for it.
[0008] The simplified ASP web page file must be changed into the script which may be interpreted with a script engine in the execution time. Typically, a script engine executes continuation or various declaration type commands [in / synchronous / an ASP file], in order to attain the result of a request. It is compiled as a file in which executive operation is possible, and, generally the script file performed with the script engine takes time amount as compared with the stored file. This is because a script engine not only performs a file, but must achieve an interpretation function.
[0009] Or one problem which is at the time of compiling a script file to the file in which executive operation is possible probably has the combination of various language, I hear that there is the possibility and it is in a script file. For example, a script file may contain other components written by the component written in HTML, and Visual Basic. Although various processings are used in order that a script engine may interpret these elements at the execution time, the compiler for translating a different language component into one language, i.e., one resource code file, does not exist. Furthermore, in the present server side application framework, programming required to manage dynamically client side user interface elements (for example, a text box, a list box, a carbon button, a hyperlink, an image, voice, etc.) within server side application needs a still advanced programming technique and considerable efforts. It becomes impossible for a script engine to be able to continue filling this demand continuously as these servers side process becomes more complicated.
[0010] This invention was made for these and other reasons.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

TECHNICAL FIELD

[Field of the Invention] Especially this invention relates to the server side code creation which creates the control object which generally processes the client side user interface element of a web page about a web server framework.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

WRITTEN AMENDMENT

---------- [procedure amendment]
[Filing Date] June 12, Heisei 13 (2001. 6.12)
[Procedure amendment 1]
[Document to be Amended] Description
[Item(s) to be Amended] 0107
[Method of Amendment] Modification
[Proposed Amendment]
[0107]
[Effect of the Invention] As mentioned above, according to this invention, intermediate language or a resource code file can be created from a server side resource, and a resource code file is compiled by the class in which executive operation is possible. Quick generation of the web page control object which performs the server side function containing the rendering of a client response by the class in which executive operation is possible is attained.
[Procedure amendment 2]
[Document to be Amended] Description
[Item(s) to be Amended] Explanation of a sign
[Method of Amendment] Modification
[Proposed Amendment]
[Description of Notations]
100 Client
102 Browser
104 Web Page
106 User Name
108 Addition
110 Deletion
112 HTTP Response
114 HTTP Request
116,300 Web server
118 304 HTTP pipeline
120 Handler
122 Static Contents Resource
124 Dynamic Contents Resource
126 Server Side Control Class Library
128 Client Side Control Class Library
130 330 Non-UI server component
308 Page Factory
310 ASP+ Resource

312 Control Class Library
314 Page Object
318 Text Box Object
320 322 Carbon button object
500 Computer
502 CPU
504 Memory
508 ROM
510 RAM
512 Hard Disk Drive
514 Magnetic Disk Drive
516 Removable Storage
518 Optical Disk Drive
519 Optical Disk
520, 522, 524 I/F
526 Operating System
528 Application Program
530 Program Module
532 Program Data
534 Keyboard
536 Mouse
540 Serial Port Interface
542 Monitor
544 Video Adapter
546 Remote Computer
548 LAN
550 WAN
552 Network Adaptor
554 Modem

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

EFFECT OF THE INVENTION

[Effect of the Invention]

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]
[Drawing 1] The web server which generates dynamically the web page contents displayed on the client in an operation gestalt with this invention is shown.
[Drawing 2] The flow chart of processing for processing of a client side user interface element and a rendering is shown using the server side control object in an operation gestalt with this invention.
[Drawing 3] An example of the module of a web server used with an operation gestalt with this invention is shown.
[Drawing 4] An example of the dynamic contents file (for example, ASP+ page) in an operation gestalt with this invention is shown.
[Drawing 5] An example of a useful system is shown in performing an operation gestalt with this invention.
[Drawing 6] The process flowchart showing processing of the page object in an operation gestalt with this invention is shown.
[Drawing 7] The process flowchart showing syntax analysis of the server side application for creating a data model is shown.
[Drawing 8] The process flowchart showing the traverse of the data model for creating a resource code file is shown.
[Drawing 9] The process flowchart showing the traverse of the data model for creating the resource code file in another operation gestalt of this invention is shown.
[Drawing 10] An example of the ASP+ page by this invention the syntax of may be analyzed is shown.
[Drawing 11] An example of the resource code file created using this invention by the initial ASP+ file shown in drawing 10 is shown.
[Description of Notations]

[Translation done.]